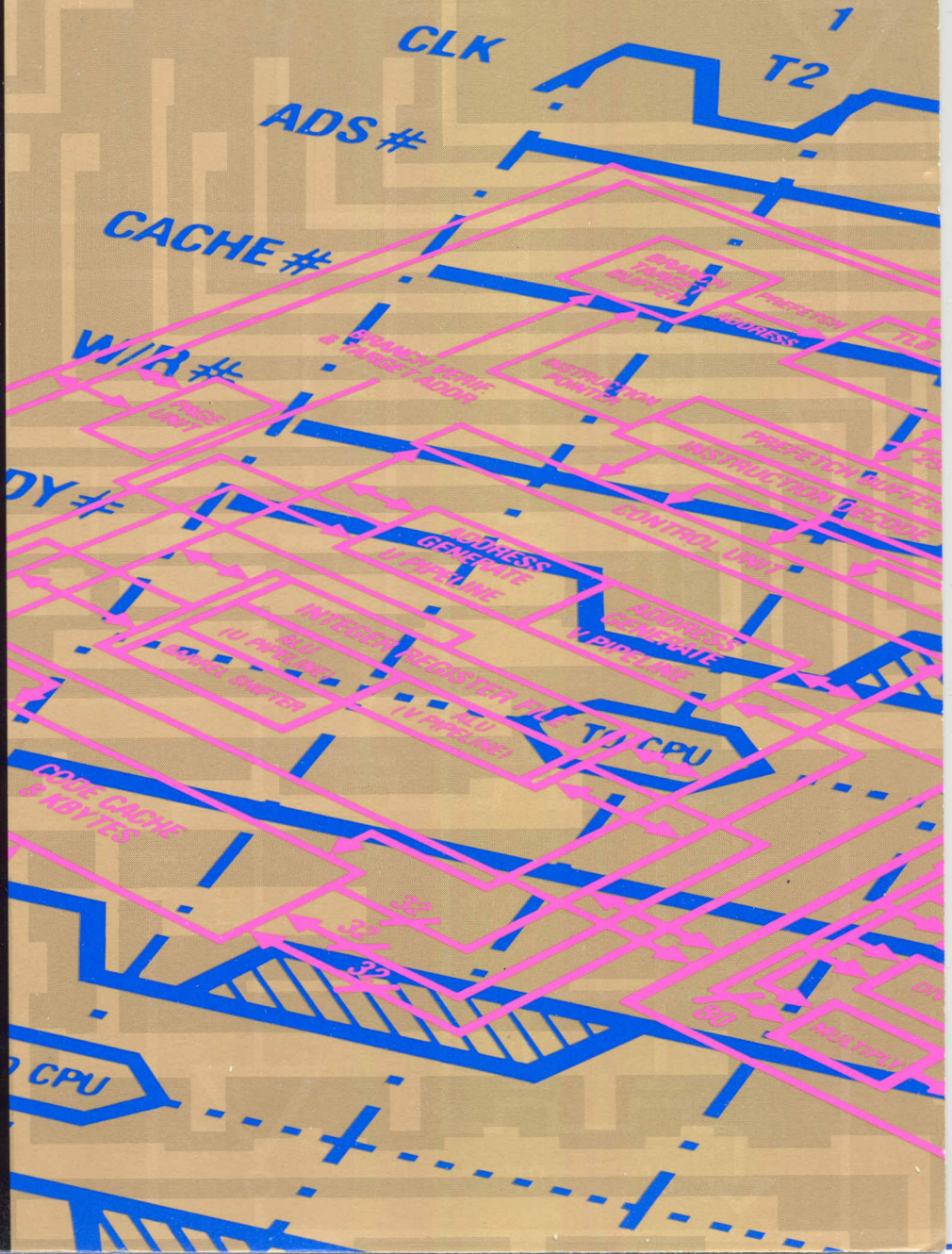


Microprocessors: Volume III

PentiumTM Processors





LITERATURE

To order Intel literature or obtain literature pricing information in the U.S. and Canada call or write Intel Literature Sales. In Europe and other international locations, please contact your **local** sales office or distributor.

INTEL LITERATURE SALES
P.O. Box 7641
Mt. Prospect, IL 60056-7641

In the U.S. and Canada
call toll free
(800) 548-4725

This 800 number is for external customers only.

CURRENT HANDBOOKS

Product line handbooks contain data sheets, application notes, article reprints and other design information. All handbooks can be ordered individually, and most are available in a pre-packaged set in the U.S. and Canada.

Title	Intel Order Number	ISBN
SET OF FOURTEEN HANDBOOKS (Available in U.S. and Canada)	231003	N/A
CONTENTS LISTED BELOW FOR INDIVIDUAL ORDERING:		
CONNECTIVITY	231658	1-55512-202-7
EMBEDDED MICROCONTROLLERS	270646	1-55512-203-5
EMBEDDED MICROPROCESSORS	272396	1-55512-204-3
FLASH MEMORY (2 volume set)	210830	1-55512-214-0
MICROPROCESSORS, VOL. 1: Intel386™ 80286 & 8086 MICROPROCESSORS	230843	1-55512-196-9
MICROPROCESSORS, VOL. 2: Intel486™ MICROPROCESSORS	241731	1-55512-197-7
MICROPROCESSORS, VOL. 3: PENTIUM™ PROCESSORS	241732	1-55512-198-5
i750®, i860™, i960® PROCESSORS AND RELATED PRODUCTS	272084	1-55512-217-5
OEM BOARDS, SYSTEMS & SOFTWARE	280407	1-55512-201-9
PACKAGING	240800	1-55512-208-6
PERIPHERAL COMPONENTS	296467	1-55512-207-8
PRODUCT OVERVIEW	210846	N/A
PROGRAMMABLE LOGIC	296083	1-55512-206-X
NETWORKING	297360	1-55512-220-5
ADDITIONAL LITERATURE: (Not included in handbook set)		
AUTOMOTIVE PRODUCTS	231792	1-55512-212-4
COMPONENTS QUALITY/RELIABILITY	210997	1-55512-132-2
CUSTOMER LITERATURE GUIDE	210620	N/A
EMBEDDED APPLICATIONS (1993/94)	270648	1-55512-179-9
INTERNATIONAL LITERATURE GUIDE (Available in Europe only)	E00029	N/A
MILITARY AND SPECIAL PRODUCTS (2 volume set)	210461	1-55512-213-2
SYSTEMS QUALITY/RELIABILITY	231762	1-55512-046-6

Microprocessors: Volume III

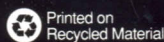
In 1993, Intel introduced the Pentium™ Processor, the fifth-generation processor based on the Intel Architecture. The Pentium processor represents the next generation of power for high-end desktop and server performance. The Pentium processor is fully code compatible with previous members of the Intel Architecture family of microprocessors, thus preserving the value of user's software investments. Up to five times as powerful as the 33-MHz Intel486™ DX CPU, the Pentium processor extends the Intel processor performance continuum while maintaining full compatibility with existing software.

The Pentium processor employs the most advanced technology and engineering innovation and is the enabling technology for today's high-end and tomorrow's emerging applications. The Pentium processor incorporates a superscalar architecture, improved floating point unit, separate on-chip code and data caches, 64-bit external data bus, and other features designed to provide an architectural platform for high-performance computing.

This handbook contains information regarding the design of Pentium processor-based systems, including secondary cache subsystems, local bus systems based on the PCI specification, and Advanced Programmable Interrupt Controller (APIC) designs.



Order Number: 241732-001
Printed in USA/194/30K/RRD/JW
Microprocessors



ISBN 1-55512-198-5





MICROPROCESSORS

VOLUME III

1994

Intel Corporation makes no warranty for the use of its products and assumes no responsibility for any errors which may appear in this document nor does it make a commitment to update the information contained herein.

Intel retains the right to make changes to these specifications at any time without notice.

Contact your local Intel sales office or your distributor to obtain the latest specifications before placing your product order.

*Other brands and names are the property of their respective owners.

Additional copies of this document or other Intel literature may be obtained from:

Intel Corporation
Literature Sales
P.O. Box 7841
Mt. Prospect, IL 60056-7841
Or call 1-800-528-4663

©1994 INTEL CORPORATION



MICROPROCESSORS VOLUME III

Intel Corporation makes no warranty for the use of its products and assumes no responsibility for any errors which may appear in this document nor does it make a commitment to update the information contained herein.

Intel retains the right to make changes to these specifications at any time, without notice.

Contact your local Intel sales office or your distributor to obtain the latest specifications before placing your product order.

MDS is an ordering code only and is not used as a product name or trademark of Intel Corporation.

Intel Corporation and Intel's FASTPATH are not affiliated with Kinetics, a division of Excelan, Inc. or its FASTPATH trademark or products.

*Other brands and names are the property of their respective owners.

Additional copies of this document or other Intel literature may be obtained from:

Intel Corporation
Literature Sales
P.O. Box 7641
Mt. Prospect, IL 60056-7641

or call 1-800-879-4683

©INTEL CORPORATION, 1993

LGCPY1/100693

DATA SHEET DESIGNATIONS

Intel uses various data sheet markings to designate each phase of the document as it relates to the product. The marking appears in the upper, right-hand corner of the data sheet. The following is the definition of these markings:

Data Sheet Marking	Description
Product Preview	Contains information on products in the design phase of development. Do not finalize a design with this information. Revised information will be published when the product becomes available.
Advanced Information	Contains information on products being sampled or in the initial production phase of development.*
Preliminary	Contains preliminary information on new products in production.*
No Marking	Contains information on products in full production.*

*Specifications within these data sheets are subject to change without notice. Verify with your local Intel sales office that you have the latest data sheet before finalizing a design.

DATA SHEET DESIGNATIONS

Intel uses various data sheet markings to designate each phase of the document as it relates to the product. The marking appears in the upper, right-hand corner of the data sheet. The following is the definition of these markings:

Data Sheet Marking	Description
Product Preview	Contains information on products in the design phase of development. Do not finalize a design with this information. Revised information will be published when the product becomes available.
Advanced Information	Contains information on products being sampled or in the initial production phase of development.
Preliminary	Contains preliminary information on new products in production.*
No Marking	Contains information on products in full production.*

*Specifications within these data sheets are subject to change without notice. Verify with your local Intel sales office that you have the latest data sheet before finalizing a design.

Overview

1

Data Sheets

2

Application Notes

3

Table of Contents

Alphanumeric Index	x
CHAPTER 1	
Overview	
The Intel Pentium™ Processor: A Technical Overview	1-1
CHAPTER 2	
Data Sheets	
Pentium Processor	2-1
OverDrive™ Processor for Pentium CPU-based Systems Socket Specification	2-32
Reference Sheet	2-43
82496 Cache Controller and 82491 Cache SRAM for Use with the Pentium	
Processor	2-44
82430 PCIset for the Pentium Processor	2-45
82489DX Advanced Programmable Interrupt Controller (APIC)	2-58
CHAPTER 3	
Application Notes	
AP-388 82489DX Programming Model	3-1
AP-478 An Example Memory Subsystem for the Pentium Processor	3-27
AP-479 Pentium Processor Clock Design	3-131
AP-480 Pentium Processor Thermal Design Guidelines	3-169
AP-481 Design with the Pentium Processor, 82496 Cache Controller, and 82491	
Cache SRAM CPU-Cache Chipset	3-196
AP-485 Intel Processor Identification with the CPUID Instruction	3-342
AB-53 The 85C224 Clock Driver Low Output Skew PLD	3-359

Alphanumeric Index

82430 PCIset for the Pentium Processor	2-45
82489DX Advanced Programmable Interrupt Controller (APIC)	2-58
82496 Cache Controller and 82491 Cache SRAM for Use with the Pentium Processor	2-44
AB-53 The 85C224 Clock Driver Low Output Skew PLD	3-359
AP-388 82489DX Programming Model	3-1
AP-478 An Example Memory Subsystem for the Pentium Processor	3-27
AP-479 Pentium Processor Clock Design	3-131
AP-480 Pentium Processor Thermal Design Guidelines	3-169
AP-481 Design with the Pentium Processor, 82496 Cache Controller, and 82491 Cache SRAM CPU-Cache Chipset	3-196
AP-485 Intel Processor Identification with the CPUID Instruction	3-342
OverDrive Processor for Pentium CPU-based Systems Socket Specification	2-32
Pentium Processor	2-1
Reference Sheet	2-43
The Intel Pentium Processor: A Technical Overview	1-1

↑

Overview

↑



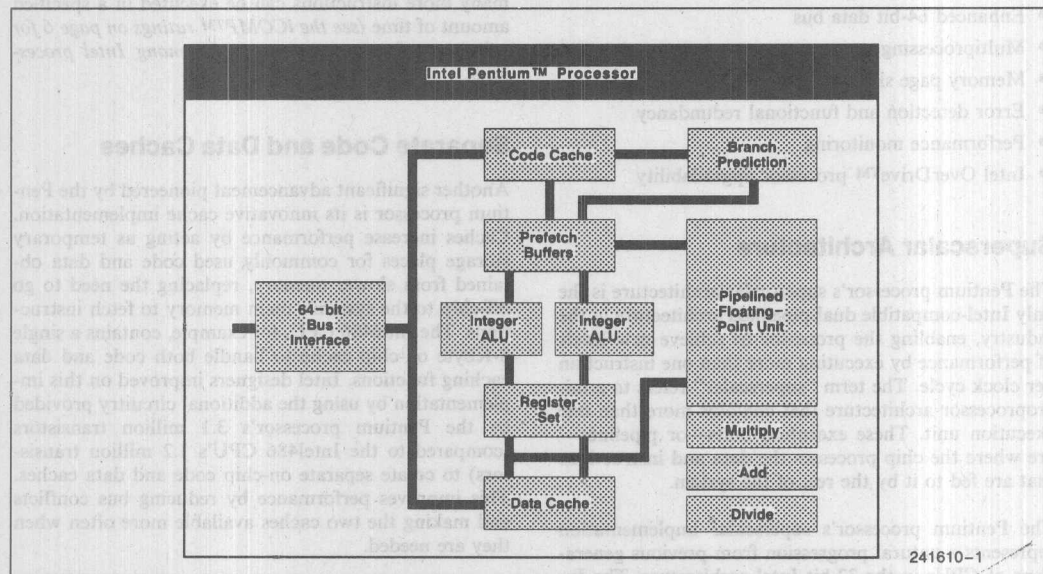
THE INTEL Pentium™ PROCESSOR A TECHNICAL OVERVIEW

Intel's new Pentium processor combines the performance traditionally associated with minicomputers and workstations with the flexibility and compatibility that characterize the personal computer platform. Designed to meet the needs of today's and tomorrow's sophisticated software applications, the Pentium processor extends the range of Intel's microprocessor architecture to new heights, blurring previous distinctions between hardware platforms and creating an entirely new realm of possibilities for desktop computers and servers.

This paper begins by presenting an overview of the Pentium processor, after which it provides details on the key technological features that enable this new Intel solution to meet the market's evolving requirements for high performance, continued software compatibility and advanced functionality. All features are explored in the context of how they benefit today's high-end desktop computer and server users, as well as those users who can now migrate from other platforms to take advantage of the Pentium processor's performance advantages.

INTEL'S NEXT GENERATION OF POWER

Incorporating more than 3.1 million transistors on a single piece of silicon, the 32-bit Pentium processor features high-performance clock speeds of 60 MHz and 66 MHz. Its superscalar architecture utilizes advanced design techniques that enable it to execute more than one instruction per clock cycle, resulting in the ability to run today's huge base of PC-compatible software faster than any other microprocessor. Beyond this existing software base, the Pentium processor's high-performance floating-point unit provides the advanced computing power necessary to tackle demanding technical and scientific applications previously reserved for the workstation platform. And as local-area and wide-area networks (LANs and WANs) continue to replace the mainframe-driven hierarchical networks of the past, the Pentium processor's multiprocessing performance and operating system flexibility are ideal for the host of new client/server applications that are appearing across the industry.





While the Pentium processor can achieve performance levels equal to or better than that of today's high-end workstations, it enjoys an advantage that none of its workstation counterparts can claim: complete compatibility with the more than 50,000 software applications—worth billions of dollars—that have been written for the Intel architecture. In addition, the Pentium processor runs all of the advanced operating systems now available for today's desktop PCs, workstations and servers, including UNIX*, Windows* and OS/2*. This provides users and systems manufacturers with the flexibility that today's heterogeneous networked distributed computing environments require.

THE PENTIUM PROCESSOR: TECHNICAL INNOVATIONS

A number of innovative product features contribute to the Pentium processor's unique combination of high performance, compatibility, data integrity and upgradability. These include:

- Superscalar architecture
- Separate code and data caches
- Branch prediction
- High-performance floating-point unit
- Enhanced 64-bit data bus
- Multiprocessing support
- Memory page size option
- Error detection and functional redundancy
- Performance monitoring
- Intel OverDrive™ processor upgradability

Superscalar Architecture

The Pentium processor's superscalar architecture is the only Intel-compatible dual-pipelined architecture in the industry, enabling the processor to achieve new levels of performance by executing more than one instruction per clock cycle. The term "superscalar" refers to a microprocessor architecture that contains more than one execution unit. These execution units—or pipelines—are where the chip processes the data and instructions that are fed to it by the rest of the system.

The Pentium processor's superscalar implementation represents a natural progression from previous generations of CPUs in the 32-bit Intel architecture. The Intel486™ CPU, for example, was able to execute many of its instructions in one clock cycle, while previous generations of Intel microprocessors required multiple clock cycles to execute a single instruction.

This ability to execute multiple instructions per clock cycle is due to the fact that the Pentium processor's two pipelines can execute two instructions simultaneously. As with the Intel486 CPU's single pipeline, the Pentium processor's dual pipelines execute integer instructions in five stages: *prefetch*, *decode 1*, *decode 2*, *execute* and *writeback*. This permits several instructions to be in various stages of execution, thus increasing processing performance. Each pipeline has its own arithmetic logic unit (ALU), address generation circuitry and data cache interface.

As with the Intel486 CPU, the Pentium processor uses hardwired instructions to replace many of the micro-coded instructions used in previous microprocessor generations. These instructions include loads, stores, and simple ALU operations which can be executed by the processor's hardware without requiring microcode. This improves performance without affecting compatibility. In the case of more complex instructions, the Pentium processor's enhanced microcode further boosts performance by employing both of the superscalar architecture's dual integer pipelines to execute instructions.

The result of these architectural innovations is that, compared to previous microprocessor implementations, many more instructions can be executed in a specified amount of time (see the *iCOMP™* ratings on page 6 for relative performance comparisons among Intel processors).

Separate Code and Data Caches

Another significant advancement pioneered by the Pentium processor is its innovative cache implementation. Caches increase performance by acting as temporary storage places for commonly used code and data obtained from slower memory, replacing the need to go off-chip to the system's main memory to fetch instructions. The Intel486 CPU, for example, contains a single 8-Kbyte on-chip cache to handle both code and data caching functions. Intel designers improved on this implementation by using the additional circuitry provided by the Pentium processor's 3.1 million transistors (compared to the Intel486 CPU's 1.2 million transistors) to create separate on-chip code and data caches. This improves performance by reducing bus conflicts and making the two caches available more often when they are needed.

The Pentium processor's code and data caches each contain 8 Kbytes of information, and both are organized as two-way set associative caches—meaning that they save time by searching only pre-specified 32-byte segments, rather than the entire cache. All of these performance-enhancing features are in turn supplemented by the Pentium processor's 64-bit internal data bus, which ensures that the dual caches and superscalar execution pipelines are continually supplied with data.

In addition, the data cache uses two other important techniques: “writeback” caching and an algorithm called the MESI (Modified, Exclusive, Shared, Invalid) protocol. Writeback caches complete writes in the cache without going out to main memory as in previous-generation “write-through” cache implementations. This technique increases performance by reducing bus utilization and preventing needless bottlenecks in the system. And the MESI protocol ensures that data in the cache and in main memory is consistent—an important consideration in advanced multiprocessing systems where different processors often need to operate on the same data.

Branch Prediction

Branch prediction is an advanced computing technique that boosts performance by keeping execution pipelines full, based on predetermining the most likely set of instructions to be executed. The Intel Pentium processor is the first and only PC-compatible processor to use branch prediction, which until now has traditionally been associated with the mainframe computing platform.

For a better understanding of this concept, consider a typical application program. After each pass through a software loop, the program performs a conditional test to determine whether to return to the beginning of the loop or to exit and continue on to the next execution step. These two choices, or paths, are called branches. Branch prediction forecasts which branch the software will require, based on the assumption that the previous branch that was taken will be used again. The Pentium processor makes branch predictions using a branch target buffer (BTB). Unlike alternative architectures, this software-transparent innovation eliminates the need for recompiling code, thus increasing overall speed and application software performance.

High-Performance Floating-Point Unit

The emerging wave of 32-bit software applications includes many compute-intensive, graphically oriented programs that require a high degree of floating-point processing power to handle mathematical calculations. As the floating-point requirements of personal computer software have steadily increased, advances in microprocessor technology have been introduced to satisfy these needs. The Intel486 DX CPU, for example, was the first Intel microprocessor to integrate math coprocessing functions on-chip; previous-generation Intel processors used off-chip math coprocessors when floating-point calculations were required.

The Pentium processor takes math computational ability to the next performance level by using an enhanced on-chip floating-point unit that incorporates sophisticated seven-stage pipelining and hardwired functions. A three-stage floating-point instruction pipeline is appended to the integer pipelines. Most floating-point instructions begin execution in one of the integer pipelines, then move on to the floating-point pipeline. Common floating-point functions—such as add, multiply and divide—are hardwired for faster execution.

As a result of these innovations, the Pentium processor executes floating-point instructions five to ten times faster than the 33-MHz Intel486 DX CPU, optimizing it for the high-speed numeric calculations inherent in advanced visual applications such as CAD and 3D graphics.

Enhanced 64-Bit Data Bus

The data bus is the highway that carries information between the processor and the memory subsystem. Because it features a 64-bit data bus, the Pentium processor has a substantially higher bus bandwidth than the Intel486 DX CPU—528 MBytes/sec at 66 MHz, compared to 160 MB/sec at 50 MHz for the Intel486 DX microprocessor. This wider data bus facilitates high-speed processing by maintaining the flow of instructions and data to the processor’s superscalar execution unit, resulting in much higher overall performance for the Pentium processor when compared to the Intel486 CPU.

In addition to having a wider data bus, the Pentium processor implements bus cycle pipelining to increase bus bandwidth. Bus cycle pipelining allows a second cycle to start before the first one is completed. This gives the memory subsystem more time to decode the address, which allows slower and less-expensive memory components to be used—resulting in a lower overall system cost. Burst reads and writes, parity on address and data, and a simple cycle identification all contribute to providing better bandwidth and improved system reliability.

Multiprocessing

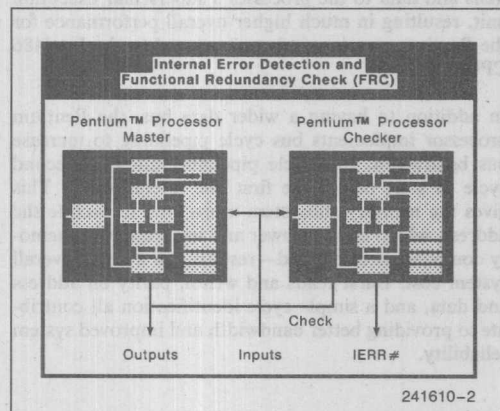
The Pentium processor is ideal for the increasing wave of multiprocessing systems and the greater levels of performance and scalability they provide for today’s computing arena. Multiprocessing applications that combine two or more Pentium processors are well served by the chip’s advanced architecture, separate on-chip code and data caches, chip sets for controlling external caches and sophisticated data integrity features.

As previously discussed, the Pentium processor maintains cache consistency with its MESI protocol. When one processor accesses data that is cached in another processor, it is ensured of receiving the correct data. And if data is modified, all processors accessing that data receive the modified version. The new Intel Pentium processor also ensures that instructions are seen by the system in the order that they were programmed. This strong ordering helps software designed to run on a single-processor system to work correctly in a multiprocessor environment.

Memory Page Size Option

The Pentium processor offers the option of supporting either the traditional memory page size of 4 KBytes, or a larger, 4-MByte page. This optional feature was provided to reduce the frequency of page swapping in complex graphics applications, frame buffers, and operating system kernels, where the increased page size now allows users to map large, previously unwieldy objects. An increased page hit rate results in higher performance, all of which is transparent to the application software.

Error Detection and Functional Redundancy



Protecting important data and ensuring its integrity throughout the entire enterprise has become increasingly important as mission-critical applications continue to proliferate across today's client/server environments. The Pentium processor incorporates two advanced features traditionally associated with mainframe-class designs—internal error detection and functional redundancy checking (FRC)—to help preserve data integrity in today's evolving desktop-based networks.

Internal error detection places parity bits on the internal code and data caches, translation look aside buffers, microcode, and branch target buffer, helping to detect errors in a manner that remains transparent to both the user and the system. Functional redundancy checking, meanwhile, is optimized for mission-critical applications where two Pentium processors can operate in a master/checker configuration. If a discrepancy is discovered between the two processors, the system is notified.

Performance Monitoring

Performance monitoring is a feature of the Pentium processor that enables system designers and application developers to optimize their hardware and software products by identifying potential code bottlenecks. Designers can observe and count clocks for internal processor events that affect the performance of data reads and writes, cache hits and misses, interrupts, and bus utilization. This allows them to measure the effect that their code has on both the Pentium processor architecture and their product and to fine-tune their application or system for optimal performance. The benefit to end users is better value and higher performance, due to the greater synergy between the Pentium processor, its host system, and application software.

Upgradability

As with all new implementations of the Intel 32-bit microprocessor architecture, the Pentium processor has been designed for easy upgradability using Intel's upgrade technology. This innovation protects user investments by adding performance that helps to maintain the productivity levels of Intel processor-based systems over their entire lifespans.

Upgrade technology makes it possible for users to take advantage of more advanced processor technology in their existing systems with an easy-to-install, single-chip performance upgrade. For example Intel's first upgrade options, the OverDrive™ processors developed for Intel486 SX and Intel486 DX CPUs, apply the same speed-doubling technology used in the development of the Intel486 DX2 microprocessor.

By installing one of these upgrade processors in the socket located next to the CPU on many Intel486 processor-based system motherboards, users can enhance overall system performance by as much as 70 percent across all software applications. And while the processor's speed is substantially increased, it still operates externally at its original clock rate, enabling it to interface seamlessly with the rest of the system and its associated peripherals.

Intel's upgrade processors will also be available for systems based on the Pentium processor family, ensuring an easy future upgrade path based on even more advanced processor technology. In addition, Pentium processor technology will be the basis of the upgrade processor now being developed for Intel486 DX2 CPU-based systems.

Increased Performance: By the Numbers

The Pentium processor stands alone on the performance ladder when compared to all other PC-compatible microprocessors, raising the standard for the Intel 32-bit architecture. While there are many ways to measure performance, three different examples offered here demonstrate the Pentium processor's speed and processing power when compared to other alternatives: the SPECint92 and SPECfp92 UNIX benchmarks, and Intel's iCOMP™ index.

SPECint92 (Figure 1) is a processor-intensive UNIX benchmark that evaluates high-end desktop performance using a representative mix of application instructions. With a SPECint92 rating of 67.4, the 66-MHz Pentium processor outperforms many workstation-class, RISC-based processors, including members of the IBM, MIPS and Sun SPARC processor families.

The SPECfp92 UNIX benchmark (Figure 2) is a useful measure of floating-point performance. The 66-MHz Pentium processor's SPECfp92 rating of 63.6 is comparable to that of today's RISC architecture and more than 3.5 times that of the Intel486 DX2-66 CPU—which previously offered the highest floating-point performance of any processor compatible with the huge base of today's industry-standard PC application software.

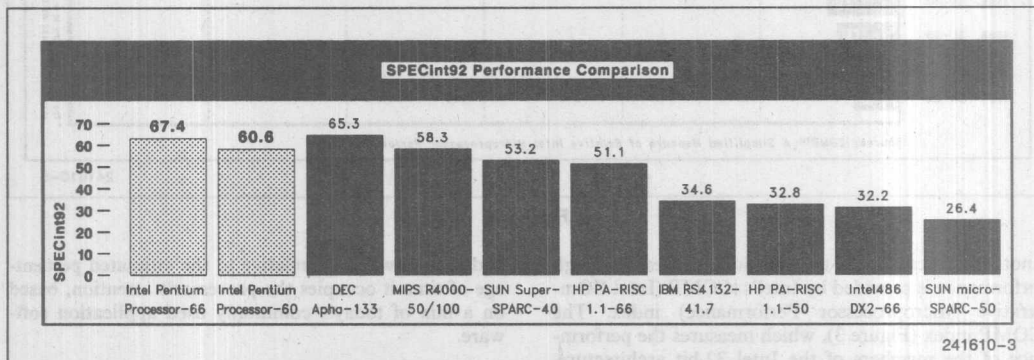


Figure 1

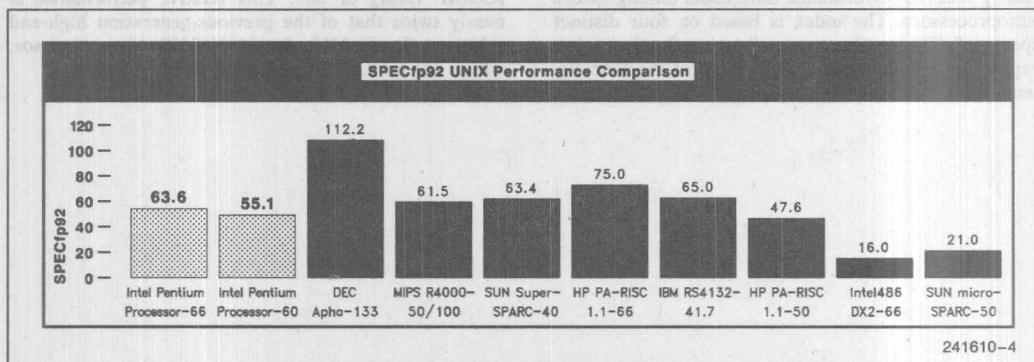


Figure 2

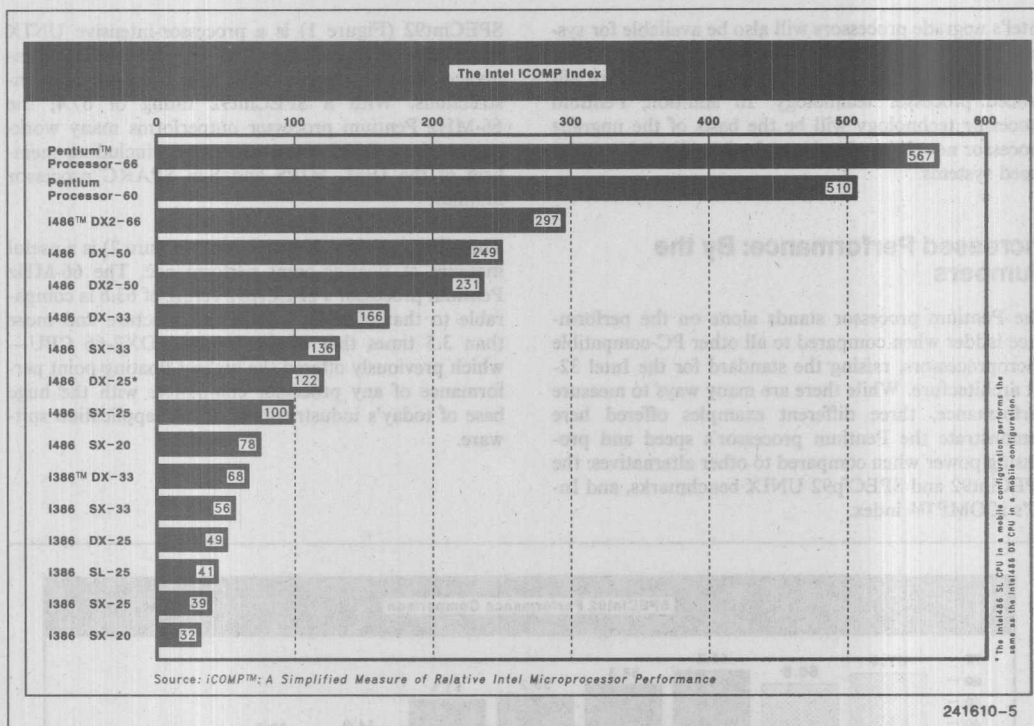


Figure 3

Another indication of the Pentium processor's high performance is provided by Intel's iCOMP (Intel Comparative Microprocessor Performance) index. The iCOMP index (Figure 3), which measures the performance of the members of the Intel 32-bit architecture, was created so that computer users can more easily identify relative performance differences among Intel's microprocessors. The index is based on four distinct aspects of CPU performance: integer, floating-point, graphics and video performance. Each of the four elements is considered for both 16-bit and 32-bit software,

and each is weighted relative to the estimated percentage of time it occupies the processor's attention, based on a mix of today's commonly used application software.

Operating at 66 MHz, the Pentium processor has an iCOMP rating of 567. This relative performance is nearly twice that of the previous-generation high-end solution, the 66-MHz Intel486 DX2 microprocessor, which has an iCOMP rating of 297.

Intel is a registered trademark of Intel Corporation.
 iCOMP™, Intel386™, Intel486™, OverDrive™ and Pentium™ are trademarks of Intel Corporation.
 OS/2™ is a trademark of International Business Machines Corporation.
 Windows™ is a trademark of Microsoft Corporation.
 UNIX® is a registered trademark of UNIX System Labs.





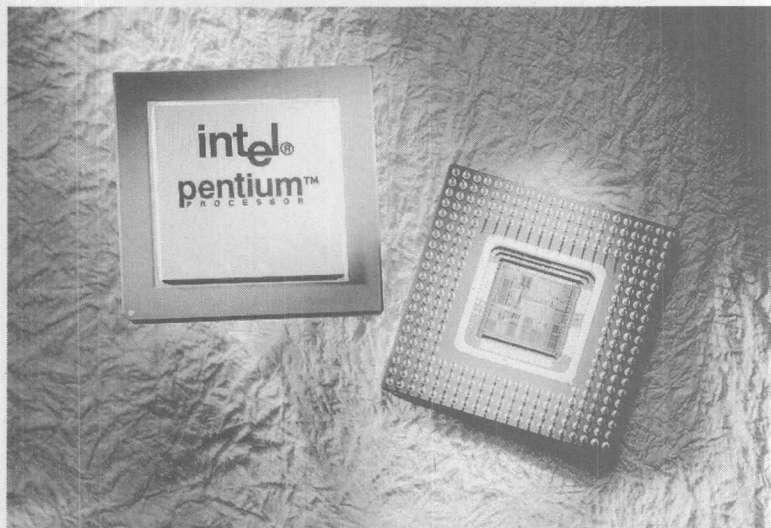
PRELIMINARY

PENTIUM™ PROCESSOR

- **Binary Compatible with Large Software Base**
 - DOS, OS/2, UNIX, and WINDOWS
- **32-Bit Microprocessor**
 - 32-Bit Addressing
 - 64-Bit Data Bus
- **Superscalar Architecture**
 - Two Pipelined Integer Units
 - Capable of under One Clock per Instruction
 - Pipelined Floating Point Unit
- **Separate Code and Data Caches**
 - 8K Code, 8K Write Back Data
 - 2-Way 32-Byte Line Size
 - Software Transparent
 - MESI Cache Consistency Protocol
- **Advanced Design Features**
 - Branch Prediction
 - Virtual Mode Extensions
- **273-Pin Grid Array Package**
- **BiCMOS Silicon Technology**
- **Increased Page Size**
 - 4M for Increased TLB Hit Rate
- **Multi-Processor Support**
 - Multiprocessor Instructions
 - Support for Second Level Cache
- **Internal Error Detection**
 - Functional Redundancy Checking
 - Built in Self Test
 - Parity Testing and Checking
- **IEEE 1149.1 Boundary Scan Compatibility**
- **Performance Monitoring**
 - Counts Occurrence of Internal Events
 - Traces Execution through Pipelines

2

The Pentium processor provides the next generation of power for high-end workstations and servers. The Pentium processor is compatible with the entire installed base of applications for DOS, Windows, OS/2, and UNIX. The Pentium processor's superscalar architecture can execute two instructions per clock cycle. Branch Prediction and separate caches also increase performance. The pipelined floating point unit of the Pentium processor delivers workstation level performance. Separate code and data caches reduce cache conflicts while remaining software transparent. The Pentium processor has 3.1 million transistors and is built on Intel's 0.8 Micron BiCMOS silicon technology.



241595-1

MS-DOS and Windows are registered trademarks of Microsoft Corporation.
OS/2 is a trademark of International Business Machines Corporation.
UNIX is a registered trademark of UNIX System Laboratories, Inc.

Pentium™ Processor

CONTENTS

PAGE

1.0 MICROPROCESSOR ARCHITECTURE OVERVIEW	2-3
2.0 PINOUT	2-6
2.1 Pinout and Pin Descriptions	2-6
2.1.1 Pentium™ Processor Pinout	2-6
2.2 Design Notes	2-9
2.3 Quick Pin Reference	2-9
2.4 Pin Reference Tables	2-16
2.5 Pin Grouping According to Function	2-18
2.6 Output Pin Grouping According to when Driven	2-18

CONTENTS

PAGE

3.0 ELECTRICAL SPECIFICATIONS	2-18
3.1 Power and Ground	2-18
3.2 Decoupling Recommendations	2-19
3.3 Connection Specifications	2-19
3.4 Maximum Ratings	2-19
3.5 D.C. Specifications	2-20
3.6 A.C. Specifications	2-20
4.0 MECHANICAL SPECIFICATIONS	2-29
5.0 THERMAL SPECIFICATIONS	2-31

The Pentium processor provides the next generation of power for high-end workstations and servers. The Pentium processor is compatible with the entire installed base of applications for DOS, Windows, OS/2, and UNIX. The Pentium processor's superscalar architecture can execute two instructions per clock cycle. Branch prediction and separate caches also increase performance. The pipelined floating point unit of the Pentium processor delivers workstation level performance. Separate code and data caches reduce cache conflicts and improve performance. The Pentium processor has 3.1 million transistors and is built on Intel's 0.5 micron silicon technology.

1.0 MICROPROCESSOR ARCHITECTURE OVERVIEW

The Pentium™ processor is the next generation member of the Intel386™ and Intel486™ microprocessor family. It is 100% binary compatible with the 8086/88, 80286, Intel386 DX CPU, Intel386 SX CPU, Intel486 DX CPU, Intel486 SX and the Intel486 DX2 CPUs.

The Pentium processor contains all of the features of the Intel486 CPU, and provides significant enhancements and additions including the following:

- Superscalar Architecture
- Dynamic Branch Prediction
- Pipelined Floating-Point Unit
- Improved Instruction Execution Time
- Separate 8K Code and Data Caches
- Writeback MESI Protocol in the Data Cache
- 64-Bit Data Bus
- Bus Cycle Pipelining
- Address Parity
- Internal Parity Checking
- Functional Redundancy Checking
- Execution Tracing
- Performance Monitoring
- IEEE 1149.1 Boundary Scan
- System Management Mode
- Virtual Mode Extensions

The application instruction set of the Pentium processor includes the complete Intel486 CPU instruction set with extensions to accommodate some of the additional functionality of the Pentium processor. All application software written for the Intel386 and Intel486 microprocessors will run on the Pentium processor without modification. The on-chip memory management unit (MMU) is completely compatible with the Intel386 and Intel486 CPUs.

The Pentium processor implements several enhancements to increase performance. The two instruction pipelines and floating-point unit on the Pentium processor are capable of independent operation. Each pipeline issues frequently used instructions in a single clock. Together, the dual pipes can issue two integer instructions in one clock, or one floating point instruction (under certain circumstances, 2 floating point instructions) in one clock.

Branch prediction is implemented in the Pentium processor. To support this, the Pentium processor implements two prefetch buffers, one to prefetch code in a linear fashion, and one that prefetches code according to the BTB so the needed code is almost always prefetched before it is needed for execution.

The floating-point unit has been completely redesigned over the Intel486 CPU. Faster algorithms provide up to 10X speed-up for common operations including add, multiply, and load.

The Pentium processor includes separate code and data caches integrated on chip to meet its performance goals. Each cache is 8 Kbytes in size, with a 32-byte line size and is 2-way set associative. Each cache has a dedicated Translation Lookaside Buffer (TLB) to translate linear addresses to physical addresses. The data cache is configurable to be write-back or writethrough on a line by line basis and follows the MESI protocol. The data cache tags are triple ported to support two data transfers and an inquire cycle in the same clock. The code cache is an inherently write protected cache. The code cache tags are also triple ported to support snooping and split line accesses. Individual pages can be configured as cacheable or non-cacheable by software or hardware. The caches can be enabled or disabled by software or hardware.

The Pentium processor has increased the data bus to 64-bits to improve the data transfer rate. Burst read and burst writeback cycles are supported by the Pentium processor. In addition, bus cycle pipelining has been added to allow two bus cycles to be in progress simultaneously. The Pentium processor Memory Management Unit contains optional extensions to the architecture which allow 4 Mbyte page sizes.

The Pentium processor has added significant data integrity and error detection capability. Data parity checking is still supported on a byte by byte basis. Address parity checking, and internal parity checking features have been added along with a new exception, the machine check exception. In addition, the Pentium processor has implemented functional redundancy checking to provide maximum error detection of the processor and the interface to the processor. When functional redundancy checking is used, a second processor, the "checker" is used to execute in lock step with the "master" processor. The

checker samples the master's outputs and compares those values with the values it computes internally, and asserts an error signal if a mismatch occurs.

As more and more functions are integrated on chip, the complexity of board level testing is increased. To address this, the Pentium processor has increased test and debug capability. Like many of the Intel486 CPUs, the Pentium processor implements IEEE Boundary Scan (Standard 1149.1). In addition, the Pentium processor has specified 4 breakpoint pins that correspond to each of the debug registers and externally indicate a breakpoint match. Execution

tracing provides external indications when an instruction has completed execution in either of the two internal pipelines, or when a branch has been taken.

System management mode has been implemented along with some extensions to the SMM architecture. Enhancements to the Virtual 8086 mode have been made to increase performance by reducing the number of times it is necessary to trap to a virtual 8086 monitor.

Figure 1-1 shows a block diagram of the Pentium processor.

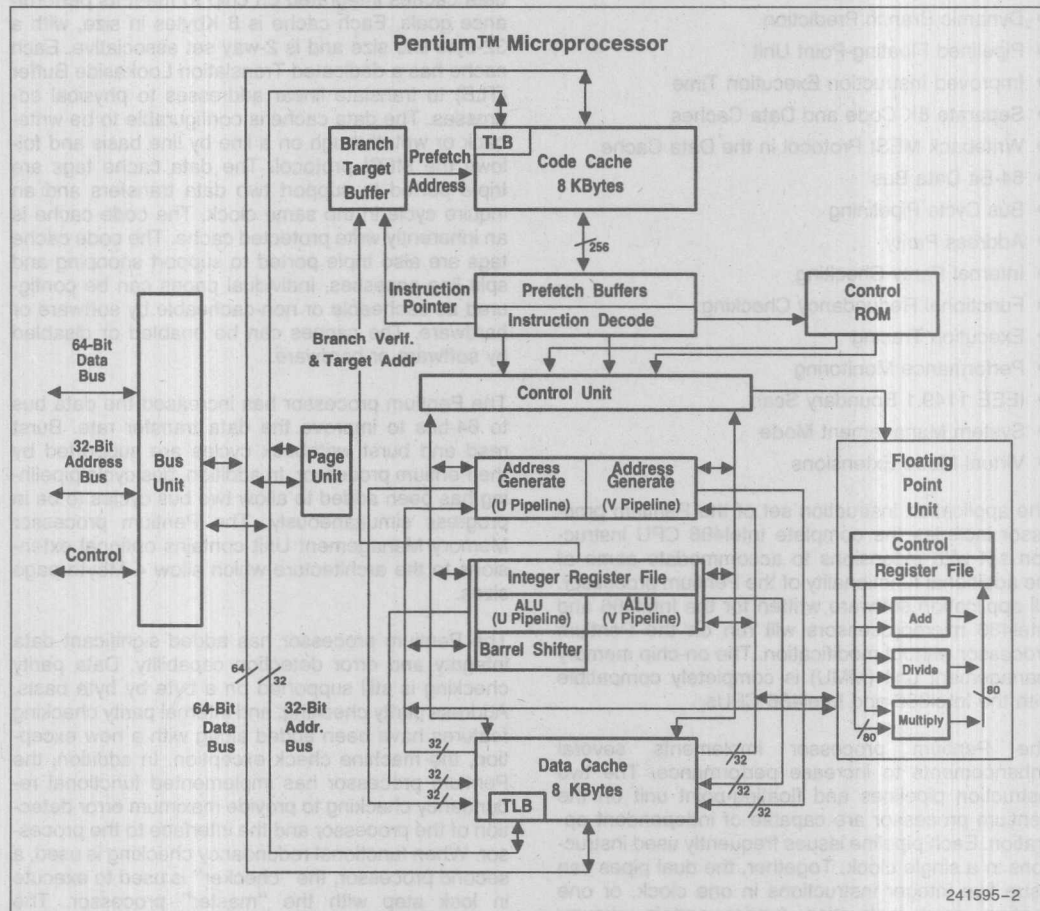


Figure 1-1. Pentium™ Processor Block Diagram

The block diagram shows the two instruction pipelines, the "u" pipe and the "v" pipe. The u-pipe can execute all integer and floating point instructions. The v-pipe can execute simple integer instructions and the FXCH floating point instructions.

The separate caches are shown, the code cache and data cache. The data cache has two ports, one for each of the two pipes (the tags are triple ported to allow simultaneous inquire cycles). The data cache has a dedicated Translation Lookaside Buffer (TLB) to translate linear addresses to the physical addresses used by the data cache.

The code cache, branch target buffer and prefetch buffers are responsible for getting raw instructions into the execution units of the Pentium processor. Instructions are fetched from the code cache or from the external bus. Branch addresses are remembered by the branch target buffer. The code

cache TLB translates linear addresses to physical addresses used by the code cache.

The decode unit decodes the prefetched instructions so the Pentium processor can execute the instruction. The control ROM contains the microcode which controls the sequence of operations that must be performed to implement the Pentium processor architecture. The control ROM unit has direct control over both pipelines.

The Pentium processor contains a pipelined floating point unit that provides a significant floating point performance advantage over previous generations of the Pentium processor.

The architectural features introduced in this chapter are more fully described in the *Pentium™ Processor User's Manual*.

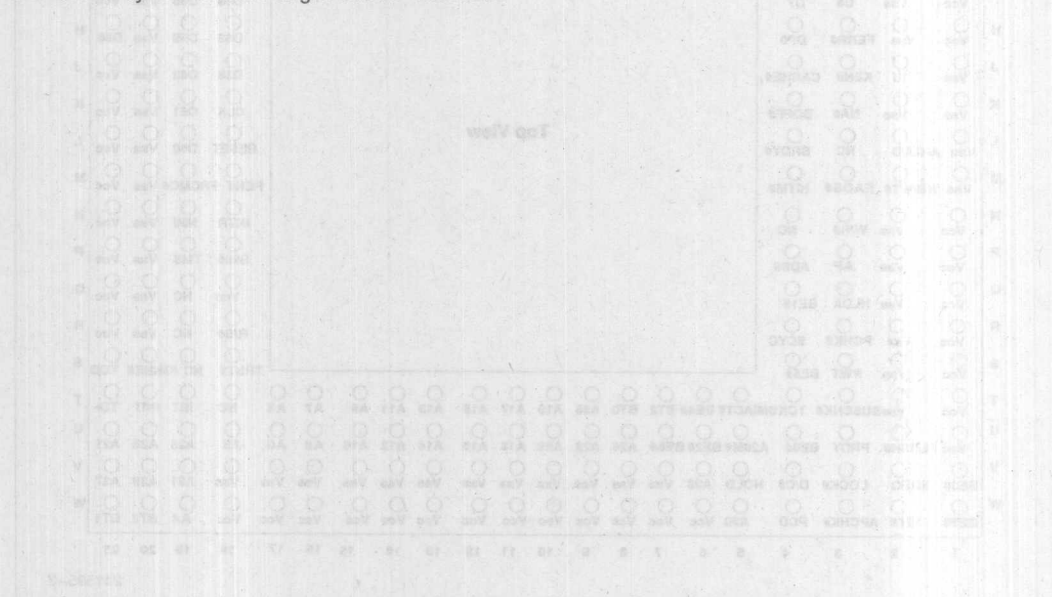


Figure 2-1. Pentium Processor Pinout (Top View)



2.0 PINOUT

2.1 Pinout and Pin Descriptions

2.1.1 Pentium™ PROCESSOR PINOUT

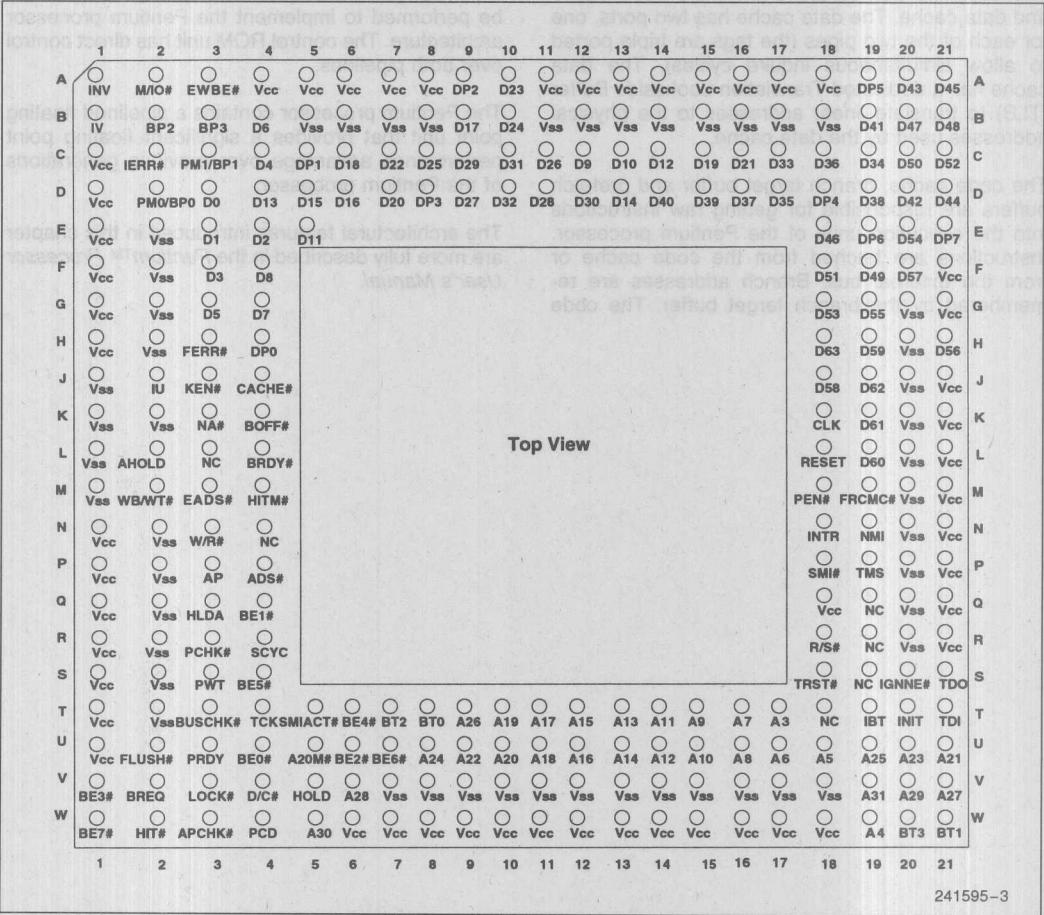


Figure 2-1. Pentium™ Processor Pinout (Top View)

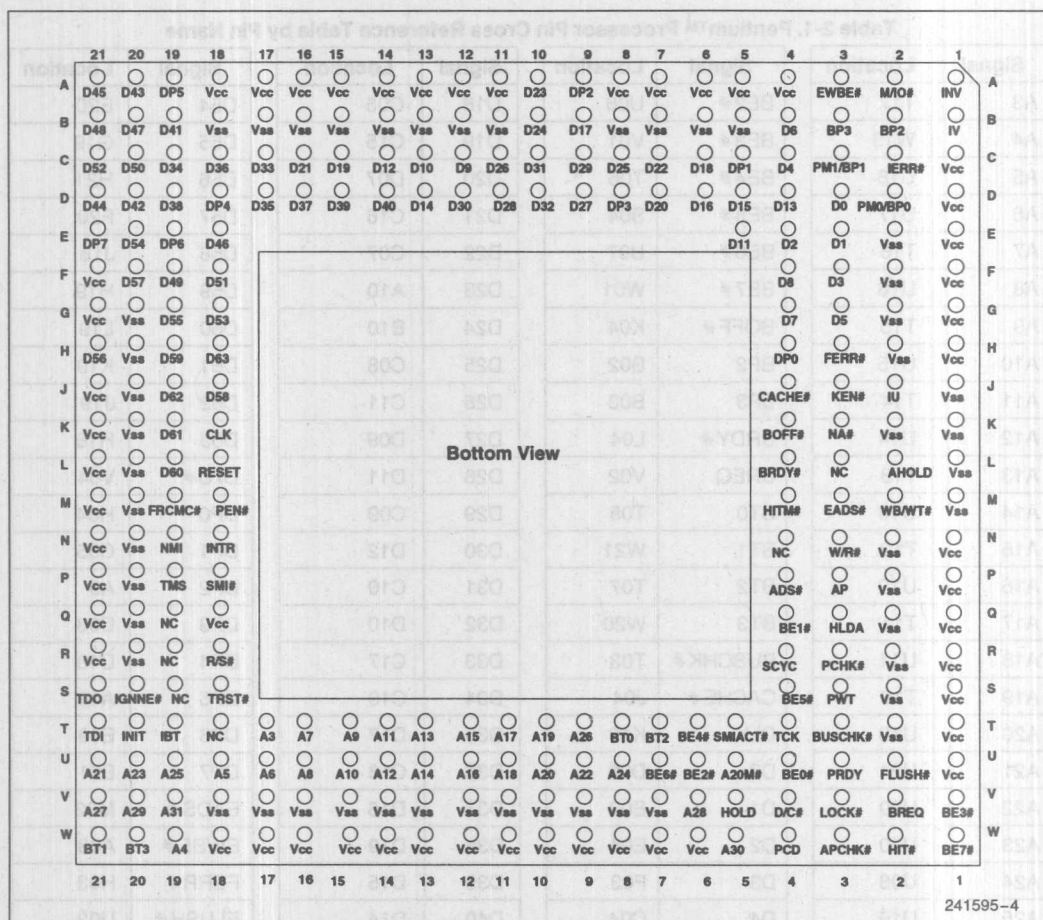


Figure 2-2. Pentium™ Processor Pinout (Bottom View)

Table 2-1. Pentium™ Processor Pin Cross Reference Table by Pin Name

Signal	Location	Signal	Location	Signal	Location	Signal	Location
A3	T17	BE2 #	U06	D18	C06	D54	E20
A4	W19	BE3 #	V01	D19	C15	D55	G19
A5	U18	BE4 #	T06	D20	D07	D56	H21
A6	U17	BE5 #	S04	D21	C16	D57	F20
A7	T16	BE6 #	U07	D22	C07	D58	J18
A8	U16	BE7 #	W01	D23	A10	D59	H19
A9	T15	BOFF #	K04	D24	B10	D60	L19
A10	U15	BP2	B02	D25	C08	D61	K19
A11	T14	BP3	B03	D26	C11	D62	J19
A12	U14	BRDY #	L04	D27	D09	D63	H18
A13	T13	BREQ	V02	D28	D11	D/C #	V04
A14	U13	BT0	T08	D29	C09	DP0	H04
A15	T12	BT1	W21	D30	D12	DP1	C05
A16	U12	BT2	T07	D31	C10	DP2	A9
A17	T11	BT3	W20	D32	D10	DP3	D08
A18	U11	BUSCHK #	T03	D33	C17	DP4	D18
A19	T10	CACHE #	J04	D34	C19	DP5	A19
A20	U10	CLK	K18	D35	D17	DP6	E19
A21	U21	D0	D03	D36	C18	DP7	E21
A22	U09	D1	E03	D37	D16	EADS #	M03
A23	U20	D2	E04	D38	D19	EWBE #	A03
A24	U08	D3	F03	D39	D15	FERR #	H03
A25	U19	D4	C04	D40	D14	FLUSH #	U02
A26	T09	D5	G03	D41	B19	FRCMC #	M19
A27	V21	D6	B04	D42	D20	HIT #	W02
A28	V06	D7	G04	D43	A20	HITM #	M04
A29	V20	D8	F04	D44	D21	HLDA	Q03
A30	W05	D9	C12	D45	A21	HOLD	V05
A31	V19	D10	C13	D46	E18	IBT	T19
A20M #	U05	D11	E05	D47	B20	IERR #	C02
ADS #	P04	D12	C14	D48	B21	IGNNE #	S20
AHOLD	L02	D13	D04	D49	F19	INIT	T20
AP	P03	D14	D13	D50	C20	INTR	N18
APCHK #	W03	D15	D05	D51	F18	INV	A01
BE0 #	U04	D16	D06	D52	C21	IU	J02
BE1 #	Q04	D17	B09	D53	G18	IV	B01

Table 2-1. Pentium™ Processor Pin Cross Reference Table by Pin Name (Continued)

Signal	Location	Signal	Location	Signal	Location	Signal	Location
KEN#	J03	RESET	L18	NC	L03, N04, Q19, R19, S19, T18	V _{SS}	B05, B06, B07, B08, B11, B12, B13, B14, B15, B16, B17, B18, E02, F02, G02, G20, H02, H20, J01, J20, K01, K02, K20, L01, L20, M01, M20, N02, N20, P02, P20, Q02, Q20, R02, R20, S02, T02, V07, V08, V09, V10, V11, V12, V13, V14, V15, V16, V17, V18
LOCK#	V03	R/S#	R18	V _{CC}	A04, A05, A06, A07, A08, A11, A12, A13, A14, A15, A16, A17, A18, C01, D01, E01, F01, F21, G01, G21, H01, J21, K21, L21, M21, N01, N21, P01, P21, Q01, Q18, Q21, R01, R21, S01, T01, U01, W06, W07, W08, W09, W10, W11, W12, W13, W14, W15, W16, W17, W18		
M/IO#	A02	SCYC	R04				
NA#	K03	SMI#	P18				
NMI	N19	SMIACT#	T05				
PCD	W04	TCK	T04				
PCHK#	R03	TDI	T21				
PEN#	M18	TD0	S21				
PM0/BP0	D02	TMS	P19				
PM1/BP1	C03	TRST#	S18				
PRDY	U03	WB/WT#	M02				
PWT	S03	W/R#	N03				

2.2 Design Notes

For reliable operation, always connect unused inputs to an appropriate signal level. Unused active LOW inputs should be connected to V_{CC}. Unused active HIGH inputs should be connected to GND.

No Connect (NC) pins must remain unconnected. Connection of NC pins may result in component failure or incompatibility with processor steppings.

NOTE:

The No Connect pin located at L03 (BRDYC#) along with BUSCHK# are sampled by the Pentium processor at RESET to configure the I/O buffers of the processor for use with the 82496 Cache Controller/82491 Cache SRAM secondary cache as a chip set (refer to the *82496 Cache Controller/82491 Cache SRAM Data Book for Use with the Pentium™ Processor* for further information).

2.3 Quick Pin Reference

This section gives a brief functional description of each of the pins. For a detailed description, see the Hardware Interface chapter in the *Pentium™ Processor User's Manual*, Vol. 1. **Note that all input pins must meet their AC/DC specifications to guarantee proper functional behavior.** In this section, the pins are arranged in alphabetical order. The functional grouping of each pin is listed at the end of this chapter.

The # symbol at the end of a signal name indicates that the active, or asserted state occurs when the signal is at a low voltage. When a # symbol is not present after the signal name, the signal is active, or asserted at the high voltage level.

Table 2-2. Quick Pin Reference

Symbol	Type*	Name and Function
A20M#	I	When the <i>address bit 20 mask</i> pin is asserted, the Pentium™ Processor emulates the address wraparound at one Mbyte which occurs on the 8086. When A20M# is asserted, the Pentium processor masks physical address bit 20 (A20) before performing a lookup to the internal caches or driving a memory cycle on the bus. The effect of A20M# is undefined in protected mode. A20M# must be asserted only when the processor is in real mode.
A31–A3	I/O	As outputs, the <i>address</i> lines of the processor along with the byte enables define the physical area of memory or I/O accessed. The external system drives the inquire address to the processor on A31–A5.
ADS#	O	The <i>address status</i> indicates that a new valid bus cycle is currently being driven by the Pentium processor.
AHOLD	I/O	In response to the assertion of <i>address hold</i> , the Pentium processor will stop driving the address lines (A31–A3), and AP in the next clock. The rest of the bus will remain active so data can be returned or driven for previously issued bus cycles.
AP	I/O	<i>Address parity</i> is driven by the Pentium processor with even parity information on all Pentium processor generated cycles in the same clock that the address is driven. Even parity must be driven back to the Pentium processor during inquire cycles on this pin in the same clock as EADS# to ensure that the correct parity check status is indicated by the Pentium processor.
APCHK#	O	The <i>address parity check</i> status pin is asserted two clocks after EADS# is sampled active if the Pentium processor has detected a parity error on the address bus during inquire cycles. APCHK# will remain active for one clock each time a parity error is detected.
BE7#–BE0#	O	The <i>byte enable</i> pins are used to determine which bytes must be written to external memory, or which bytes were requested by the CPU for the current cycle. The byte enables are driven in the same clock as the address lines (A31–3).
BOFF#	I	The <i>backoff</i> input is used to abort all outstanding bus cycles that have not yet completed. In response to BOFF#, the Pentium processor will float all pins normally floated during bus hold in the next clock. The processor remains in bus hold until BOFF# is negated at which time the Pentium processor restarts the aborted bus cycle(s) in their entirety.
BP[3:2] PM/BP[1:0]	O	The <i>breakpoint</i> pins (BP3–0) correspond to the debug registers, DR3–DR0. These pins externally indicate a breakpoint match when the debug registers are programmed to test for breakpoint matches. BP1 and BP0 are multiplexed with the Performance Monitoring pins (PM1 and PM0). The PB1 and PB0 bits in the Debug Mode Control Register determine if the pins are configured as breakpoint or performance monitoring pins. The pins come out of reset configured for performance monitoring.
BRDY#	I	The <i>burst ready</i> input indicates that the external system has presented valid data on the data pins in response to a read or that the external system has accepted the Pentium processor data in response to a write request. This signal is sampled in the T2, T12 and T2P bus states.
BREQ	O	The <i>bus request</i> output indicates to the external system that the Pentium processor has internally generated a bus request. This signal is always driven whether or not the Pentium processor is driving its bus.
BT3–BT0	O	The <i>branch trace</i> outputs provide bits 2-0 of the branch target linear address (BT2–BT0) and the default operand size (BT3) during a branch trace message special cycle.

Table 2-2. Pin reference (continued)

Symbol	Type*	Name and Function
BUSCHK#	I	The <i>bus check</i> input allows the system to signal an unsuccessful completion of a bus cycle. If this pin is sampled active, the Pentium processor will latch the address and control signals in the machine check registers. If in addition, the MCE bit in CR4 is set, the Pentium processor will vector to the machine check exception.
CACHE#	O	For Pentium processor-initiated cycles the <i>cache</i> pin indicates internal cacheability of the cycle (if a read), and indicates a burst writeback cycle (if a write). If this pin is driven inactive during a read cycle, Pentium processor will not cache the returned data, regardless of the state of the KEN# pin. This pin is also used to determine the cycle length (number of transfers in the cycle).
CLK	I	The <i>clock</i> input provides the fundamental timing for the Pentium processor. Its frequency is the internal operating frequency of the Pentium processor and requires TTL levels. All external timing parameters except TDI, TDO, TMS and TRST# are specified with respect to the rising edge of CLK.
D/C#	O	The <i>Data/Code</i> output is one of the primary bus cycle definition pins. It is driven valid in the same clock as the ADS# signal is asserted. D/C# distinguishes between data and code or special cycles.
D63-D0	I/O	These are the 64 <i>data lines</i> for the processor. Lines D7-D0 define the least significant byte of the data bus; lines D63-D56 define the most significant byte of the data bus. When the CPU is driving the data lines, they are driven during the T2, T12, or T2P clocks for that cycle. During reads, the CPU samples the data bus when BRDY# is returned.
DP7-DP0	I/O	These are the <i>data parity</i> pins for the processor. There is one for each byte of the data bus. They are driven by the Pentium processor with even parity information on writes in the same clock as write data. Even parity information must be driven back to the Pentium processor on these pins in the same clock as the data to ensure that the correct parity check status is indicated by the Pentium processor. DP7 applies to D63-D56, DP0 applies to D7-D0.
EADS#	I	This signal indicates that a <i>valid external address</i> has been driven onto the Pentium processor address pins to be used for an inquire cycle.
EWBE#	I	The <i>external write buffer empty</i> input, when inactive (high), indicates that a write cycle is pending in the external system. When the Pentium processor generates a write, and EWBE# is sampled inactive, the Pentium processor will hold off all subsequent writes to all E or M-state lines in the data cache until all write cycles have completed, as indicated by EWBE# being active.
FERR#	O	The <i>floating point error</i> pin is driven active when an unmasked floating point error occurs. FERR# is similar to the ERROR# pin on the Intel387™ math coprocessor. FERR# is included for compatibility with systems using DOS type floating point error reporting.
FLUSH#	I	When asserted, the <i>cache flush</i> input forces the Pentium processor to writeback all modified lines in the data cache and invalidate its internal caches. A Flush Acknowledge special cycle will be generated by the Pentium processor indicating completion of the writeback and invalidation. If FLUSH# is sampled low when RESET transitions from high to low, tristate test mode is entered.

Table 2-2. Quick Pin Reference (Continued)

Symbol	Type*	Name and Function
FRCMC#	I	<p>The <i>Functional Redundancy Checking Master/Checker</i> mode input is used to determine whether the Pentium processor is configured in master mode or checker mode. When configured as a master, the Pentium processor drives its output pins as required by the bus protocol. When configured as a checker, the Pentium processor tristates all outputs (except IERR# and TDO) and samples the output pins.</p> <p>The configuration as a master/checker is set after RESET and may not be changed other than by a subsequent RESET.</p>
HIT#	O	The <i>hit</i> indication is driven to reflect the outcome of an inquire cycle. If an inquire cycle hits a valid line in either the Pentium processor data or instruction cache, this pin is asserted two clocks after EADS# is sampled asserted. If the inquire cycle misses Pentium processor cache, this pin is negated two clocks after EADS#. This pin changes its value only as a result of an inquire cycle and retains its value between the cycles.
HITM#	O	The <i>hit to a modified line</i> output is driven to reflect the outcome of an inquire cycle. It is asserted after inquire cycles which resulted in a hit to a modified line in the data cache. It is used to inhibit another bus master from accessing the data until the line is completely written back.
HLDA	O	The <i>bus hold acknowledge</i> pin goes active in response to a hold request driven to the processor on the HOLD pin. It indicates that the Pentium processor has floated most of the output pins and relinquished the bus to another local bus master. When leaving bus hold, HLDA will be driven inactive and the Pentium processor will resume driving the bus. If the Pentium processor has bus cycle pending, it will be driven in the same clock that HLDA is deasserted.
HOLD	I	In response to the <i>bus hold request</i> , the Pentium processor will float most of its output and input/output pins and assert HLDA after completing all outstanding bus cycles. The Pentium processor will maintain its bus in this state until HOLD is deasserted. HOLD is not recognized during LOCK cycles. The Pentium processor will recognize HOLD during reset.
IBT	O	The <i>instruction branch taken</i> pin is driven active (high) for one clock to indicate that a branch was taken. This output is always driven by the Pentium processor.
IERR#	O	The <i>internal error</i> pin is used to indicate two types of errors, internal parity errors and functional redundancy errors. If a parity error occurs on a read from an internal array, the Pentium processor will assert the IERR# pin for one clock and then shutdown. If the Pentium processor is configured as a checker and a mismatch occurs between the value sampled on the pins and the corresponding value computed internally, the Pentium processor will assert IERR# two clocks after the mismatched value is returned.
IGNNE#	I	This is the <i>ignore numeric error</i> input. This pin has no effect when the NE bit in CR0 is set to 1. When the CR0.NE bit is 0, and the IGNNE# pin is asserted, the Pentium processor will ignore any pending unmasked numeric exception and continue executing floating point instructions for the entire duration that this pin is asserted. When the CR0.NE bit is 0, IGNNE# is not asserted, a pending unmasked numeric exception exists (SW.ES = 1), and the floating point instruction is one of FINIT, FCLEX, FSTENV, FSAVE, FSTSW, FSTCW, FENI, FDISI, or FSETPM, the Pentium processor will execute the instruction in spite of the pending exception. When the CR0.NE bit is 0, IGNNE# is not asserted, a pending unmasked numeric exception exists (SW.ES = 1), and the floating point instruction is one other than FINIT, FCLEX, FSTENV, FSAVE, FSTSW, FSTCW, FENI, FDISI, or FSETPM, the Pentium processor will stop execution and wait for an external interrupt.

Table 2-2. Quick Pin Reference (Continued)

Symbol	Type*	Name and Function
INIT	I	The Pentium processor <i>initialization</i> input pin forces the Pentium processor to begin execution in a known state. The processor state after INIT is the same as the state after RESET except that the internal caches, write buffers, and floating point registers retain the values they had prior to INIT. INIT may NOT be used in lieu of RESET after power-up. If INIT is sampled high when RESET transitions from high to low the Pentium processor will perform built-in self test prior to the start of program execution.
INTR	I	An active <i>maskable interrupt</i> input indicates that an external interrupt has been generated. If the IF bit in the EFLAGS register is set, the Pentium processor will generate two locked interrupt acknowledge bus cycles and vector to an interrupt handler after the current instruction execution is completed. INTR must remain active until the first interrupt acknowledge cycle is generated to assure that the interrupt is recognized.
INV	I	The <i>invalidation</i> input determines the final cache line state (S or I) in case of an inquire cycle hit. It is sampled together with the address for the inquire cycle in the clock EADS# is sampled active.
IU	O	The <i>u-pipe instruction complete</i> output is driven active (high) for 1 clock to indicate that an instruction in the u-pipeline has completed execution. This pin is always driven by the Pentium processor.
IV	O	The <i>v-pipe instruction complete</i> output is driven active (high) for one clock to indicate that an instruction in the v-pipeline has completed execution. This pin is always driven by the Pentium processor.
KEN#	I	The <i>cache enable</i> pin is used to determine whether the current cycle is cacheable or not and is consequently used to determine cycle length. When the Pentium processor generates a cycle that can be cached (CACHE# asserted) and KEN# is active, the cycle will be transformed into a burst line fill cycle.
LOCK#	O	The <i>bus lock</i> pin indicates that the current bus cycle is locked. The Pentium processor will not allow a bus hold when LOCK# is asserted (but AHOLD and BOFF# are allowed). LOCK# goes active in the first clock of the first locked bus cycle and goes inactive after the BRDY# is returned for the last locked bus cycle. LOCK# is guaranteed to be deasserted for at least one clock between back to back locked cycles.
M/IO#	O	The <i>Memory/Input-Output</i> is one of the primary bus cycle definition pins. It is driven valid in the same clock as the ADS# signal is asserted. M/IO# distinguishes between memory and I/O cycles.
NA#	I	An active <i>next address</i> input indicates that the external memory system is ready to accept a new bus cycle although all data transfers for the current cycle have not yet completed. The Pentium processor will drive out a pending cycle two clocks after NA# is asserted. The Pentium processor supports up to 2 outstanding bus cycles.
NMI	I	The <i>non-maskable interrupt</i> request signal indicates that an external non-maskable interrupt has been generated.
PCD	O	The <i>page cache disable</i> pin reflects the state of the PCD bit in CR3, the Page Directory Entry, or the Page Table Entry. The purpose of PCD is to provide an external cacheability indication on a page by page basis.
PCHK#	O	The <i>parity check</i> output indicates the result of a parity check on a data read. It is driven with parity status two clocks after BRDY# is returned. PCHK# remains low one clock for each clock in which a parity error was detected. Parity is checked only for the bytes on which valid data is returned.

Table 2-2. Quick Pin Reference (Continued)

Symbol	Type*	Name and Function
PEN #	I	The <i>parity enable</i> input (along with CR4.MCE) determines whether a machine check exception will be taken as a result of a data parity error on a read cycle. If this pin is sampled active in the clock a data parity error is detected, the Pentium processor will latch the address and control signals of the cycle with the parity error in the machine check registers. If in addition the machine check enable bit in CR4 is set to "1", the Pentium processor will vector to the machine check exception before the beginning of the next instruction.
PM/BP[1:0]B P[3:2]	O	These pins function as part of the Performance Monitoring feature. The breakpoint pins BP[1:0] are multiplexed with the Performance Monitoring pins PM[1:0]. The PB1 and PB0 bits in the Debug Mode Control Register determine if the pins are configured as breakpoint or performance monitoring pins. The pins come out of reset configured for performance monitoring.
PRDY	O	The PRDY output pin indicates that the processor has stopped normal execution in response to the R/S# pin going active, or Probe Mode being entered.
PWT	O	The <i>page write through</i> pin reflects the state of the PWT bit in CR3, the Page Directory Entry, or the Page Table Entry. The PWT pin is used to provide an external writeback indication on a page by page basis.
R/S#	I	The R/S# input is an asynchronous, edge sensitive interrupt used to stop the normal execution of the processor and place it into an idle state. A high to low transition on the R/S# pin will interrupt the processor and cause it to stop execution at the next instruction boundary.
RESET	I	<i>Reset</i> forces the Pentium processor to begin execution at a known state. All the Pentium processor internal caches will be invalidated upon the RESET. Modified lines in the data cache are not written back. FLUSH#, FRCMC# and INIT are sampled when RESET transitions from high to low to determine if tristate test mode or checker mode will be entered, or if BIST will be run.
SCYC	O	The <i>split cycle</i> output is asserted during misaligned LOCKed transfers to indicate that more than two cycles will be locked together. This signal is defined for locked cycles only. It is undefined for cycles which are not locked.
SMI #	I	The system Management Interrupt causes a system management interrupt request to be latched internally. When the latched SMI# is recognized on an instruction boundary, the processor enters System Management Mode.
SMACT #	O	An active <i>system management interrupt active</i> output indicates that the processor is operating in System Management Mode (SMM).
TCK	I	The <i>testability clock</i> input provides the clocking function for the Pentium processor boundary scan in accordance with the IEEE Boundary Scan interface (Standard 1149.1). It is used to clock state information and data into and out of the Pentium processor during boundary scan.
TDI	I	The <i>test data input</i> is a serial input for the test logic. TAP instructions and data are shifted into the Pentium processor on the TDI pin on the rising edge of TCK when the TAP controller is in an appropriate state.
TDO	O	The <i>test data output</i> is a serial output of the test logic. TAP instructions and data are shifted out of the Pentium processor on the TDO pin on the falling edge of TCK when the TAP controller is in an appropriate state.
TMS	I	The value of the <i>test mode select</i> input signal sampled at the rising edge of TCK controls the sequence of TAP controller state changes.

2.4 Pin Reference Tables

Table 2-4. Input Pins

Table 2-3. Output Pins

Name	Active Level	When Floated
ADS #	LOW	Bus Hold, BOFF #
APCHK #	LOW	
BE7 # - BE0 #	LOW	Bus Hold, BOFF #
BREQ	HIGH	
BT3-BT0	n/a	
CACHE #	LOW	Bus Hold, BOFF #
FERR #	LOW	
HIT #	LOW	
HITM #	LOW	
HLDA	HIGH	
IBT	HIGH	
IERR #	LOW	
IU	HIGH	
IV	HIGH	
LOCK #	LOW	Bus Hold, BOFF #
M/IO #, D/C #, W/R #	n/a	Bus Hold, BOFF #
PCHK #	LOW	
BP3-2, PM1/BP1, PM0/BP0	HIGH	
PRDY	HIGH	
PWT, PCD	HIGH	Bus Hold, BOFF #
SCYC	HIGH	Bus Hold, BOFF #
SMIACK #	LOW	
TDO	n/a	All states except Shift-DR and Shift-IR

NOTE:

All output and input/output pins are floated during tri-state test mode and checker mode (except IERR #).

Name	Active Level	Synchronous/ Asynchronous	Internal resistor	Qualified
A20M #	LOW	Asynchronous		
AHOLD	HIGH	Synchronous		
BOFF #	LOW	Synchronous		
BRDY #	LOW	Synchronous		Bus State T2, T12, T2P
BUSCHK #	LOW	Synchronous	Pullup	BRDY #
CLK	n/a			
EADS #	LOW	Synchronous		
EWBE #	LOW	Synchronous		BRDY #
FLUSH #	LOW	Asynchronous		
FRCMC #	LOW	Asynchronous		
HOLD	HIGH	Synchronous		
IGNNE #	LOW	Asynchronous		
INIT	HIGH	Asynchronous		
INTR	HIGH	Asynchronous		
INV	HIGH	Synchronous		EADS #
KEN #	LOW	Synchronous		First BRDY # / NA #
NA #	LOW	Synchronous		Bus State T2, TD, T2P
NMI	HIGH	Asynchronous		
PEN #	LOW	Synchronous		BRDY #
R/S #	n/a	Asynchronous	Pullup	
RESET	HIGH	Asynchronous		
SMI #	LOW	Asynchronous	Pullup	
TCK	n/a		Pullup	
TDI	n/a	Synchronous/TCK	Pullup	TCK
TMS	n/a	Synchronous/TCK	Pullup	TCK
TRST #	LOW	Asynchronous	Pullup	
WB/WT #	n/a	Synchronous		First BRDY # / NA #

Table 2-5. Input/Output Pins

Name	Active Level	When Floated	Qualified (when an input)
A31-A3	n/a	Address hold, Bus Hold, BOFF #	EADS #
AP	n/a	Address hold, Bus Hold, BOFF #	EADS #
D63-D0	n/a	Bus Hold, BOFF #	BRDY #
DP7-DP0	n/a	Bus Hold, BOFF #	BRDY #

NOTE:

All output and input/output pins are floated during tristate test mode (except TDO) and checker mode (except IERR # and TDO).

2

2.0 ELECTRICAL SPECIFICATIONS

2.1 Power and Ground

For clean on-chip power distribution, the Pentium processor has 50 Vcc power and 49 Vss (ground) inputs. Power and ground connections must be made to all external Vcc and Vss pins of the Pentium processor. On the circuit board, all Vcc pins must be connected to a Vcc plane, all Vss pins must be connected to a Vss plane.

2.5 Pin Grouping According to Function

Table 2-6 organizes the pins with respect to their function.

Table 2-6. Pin Functional Grouping

Function	Pins
Clock	CLK
Initialization	RESET, INIT
Address Bus	A31-A3, BE7#-BE0#
Address Mask	A20M#
Data Bus	D63-D0
Address Parity	AP, APCHK#
Data Parity	DP7-DP0, PCHK#, PEN#
Internal Parity Error	IERR#
System Error	BUSCHK#
Bus Cycle Definition	M/IO#, D/C#, W/R#, CACHE#, SCYC, LOCK#
Bus Control	ADS#, BRDY#, NA#
Page Cacheability	PCD, PWT
Cache Control	KEN#, WB/WT#
Cache Snooping/Consistency	AHOLD, EADS#, HIT#, HITM#, INV
Cache Flush	FLUSH#
Write Ordering	EWBE#
Bus Arbitration	BOFF#, BREQ, HOLD, HLDA
Interrupts	INTR, NMI
Floating Point Error Reporting	FERR#, IGNNE#
System Management Mode	SMI#, SMIACT#
Functional Redundancy Checking	FRCMC# (IERR#)
TAP Port	TCK, TMS, TDI, TDO, TRST#
Breakpoint/Performance Monitoring	PM0/BP0, PM1/BP1, BP3-2
Execution Tracing	BT3-BT0, IU, IV, IBT
Probe Mode	R/S#, PRDY

2.6 Output Pin Grouping According to when Driven

This section groups the output pins according to when they are driven.

Group 1

The following output pins are driven active at the beginning of a bus cycle with ADS#. A31-A3 and AP are guaranteed to remain valid until AHOLD is asserted or until the earlier of the clock after NA# or the last BRDY#. The remaining pins are guaranteed to remain valid until the earlier of the clock after NA# or the last BRDY#:

A31-A3, AP, BE7#-0#, CACHE#, M/IO#, W/R#, D/C#, SCYC, PWT, PCD.

Group 2

As outputs, the following pins are driven in T2, T12, and T2P. As inputs, these pins are sampled with BRDY#:

D63-0, DP7-0.

Group 3

These are the status output pins. They are always driven:

BREQ, HIT#, HITM#, IU, IV, IBT, BT3-BT0, PM0/BP0, PM1/BP1, BP3, BP2, PRDY, SMIACT#.

Group 4

These are the glitch free status output pins.

APCHK#, FERR#, HLDA, IERR#, LOCK#, PCHK#.

3.0 ELECTRICAL SPECIFICATIONS

3.1 Power and Ground

For clean on-chip power distribution, the Pentium processor has 50 V_{CC} (power) and 49 V_{SS} (ground) inputs. Power and ground connections must be made to all external V_{CC} and V_{SS} pins of the Pentium processor. On the circuit board, all V_{CC} pins must be connected to a V_{CC} plane. All V_{SS} pins must be connected to a V_{SS} plane.

3.2 Decoupling Recommendations

Liberal decoupling capacitance should be placed near the Pentium processor. The Pentium processor driving its large address and data buses at high frequencies can cause transient power surges, particularly when driving large capacitive loads.

Low inductance capacitors (i.e. surface mount capacitors) and interconnects are recommended for best high frequency electrical performance. Inductance can be reduced by connecting capacitors directly to the V_{CC} and V_{SS} planes, with minimal trace length between the component pads and vias to the plane. Capacitors specifically for PGA packages are also commercially available.

These capacitors should be evenly distributed among each component. Capacitor values should be chosen to ensure they eliminate both low and high frequency noise components.

3.3 Connection Specifications

All NC pins must remain unconnected.

For reliable operation, always connect unused inputs to an appropriate signal level. Unused active low inputs should be connected to V_{CC} . Unused active high inputs should be connected to ground.

3.4 Maximum Ratings

Table 3-1 is a stress rating only. Functional operation at the maximums is not guaranteed. Functional operating conditions are given in the A.C. and D.C. specification tables.

Extended exposure to the maximum ratings may affect device reliability. Furthermore, although the Pentium processor contains protective circuitry to resist damage from static electric discharge, always take precautions to avoid high static voltages or electric fields.

Table 3-1. Absolute Maximum Ratings

Case temperature under bias	-65°C to +110°C
Storage temperature	-65°C to +150°C
Voltage on any pin with respect to ground	-0.5 V_{CC} to $V_{CC} + 0.5$ (V)
Supply voltage with respect to V_{SS}	-0.5V to +6.5V

3.5 D.C. Specifications

Table 3-2 lists the D.C. specifications associated with the Pentium processor.

Table 3-2. Pentium™ Processor D.C. Specifications V_{CC} = See Notes 10, 11; T_{case} = See Notes 12, 13

Symbol	Parameter	Min	Max	Unit	Notes
V_{IL}	Input Low Voltage	-0.3	+0.8	V	TTL Level
V_{IH}	Input High Voltage	2.0	$V_{CC} + 0.3$	V	TTL Level
V_{OL}	Output Low Voltage		0.45	V	TTL Level, (1)
V_{OH}	Output High Voltage	2.4		V	TTL Level, (2)
I_{CC}	Power Supply Current		3200 2910	mA mA	66 MHz, (7), (8) 60 MHz, (7), (9)
I_{LI}	Input Leakage Current		± 15	μA	$0 \leq V_{IN} \leq V_{CC}$, (4)
I_{LO}	Output Leakage Current		± 15	μA	$0 \leq V_{OUT} \leq V_{CC}$ Tristate, (4)
I_{IL}	Input Leakage Current		-400	μA	$V_{IN} = 0.45V$, (5)
I_{IH}	Input Leakage Current		200	μA	$V_{IN} = 2.4V$, (6)
C_{IN}	Input Capacitance		15	pF	
C_O	Output Capacitance		20	pF	
$C_{I/O}$	I/O Capacitance		25	pF	
C_{CLK}	CLK Input Capacitance		8	pF	
C_{TIN}	Test Input Capacitance		15	pF	
C_{TOUT}	Test Output Capacitance		20	pF	
C_{TCK}	Test Clock Capacitance		8	pF	

NOTES:

1. Parameter measured at 4 mA load.
2. Parameter measured at 1 mA load.
4. This parameter is for input without pullup or pulldown.
5. This parameter is for input with pullup.
6. This parameter is for input with pulldown.
7. Worst case average I_{CC} for a mix of test patterns.
8. (16W max.) Typical Pentium processor supply current is 2600 mA (13W) at 66 MHz.
9. (14.6W max.) Typical Pentium processor supply current is 2370 mA (11.9W) at 60 MHz.
10. $V_{CC} = 5V \pm 5\%$ at 60 MHz.
11. $V_{CC} = 4.9V$ to $5.40V$ at 66 MHz.
12. $T_{case} = 0^\circ C$ to $+80^\circ C$ at 60 MHz.
13. $T_{case} = 0^\circ C$ to $+70^\circ C$ at 66 MHz.

3.6 A.C. Specifications

The 66 MHz and 60 MHz A.C. specifications given in Tables 3-3 and 3-4 consist of output delays, input setup requirements and input hold requirements. All A.C. specifications (with the exception of those for the TAP signals) are relative to the rising edge of the CLK input.

All timings are referenced to 1.5 volts for both "0" and "1" logic levels unless otherwise specified.

Within the sampling window, a synchronous input must be stable for correct Pentium processor operation.

Care should be taken to read all notes associated with a particular timing parameter. In addition, the following list of notes apply to the timing specification tables in general and are not associated with any one timing. They are 2, 5, 6, and 14.

Table 3-3. 66 MHz Pentium™ Processor A.C. Specifications
 $V_{CC} = 4.9V \text{ to } 5.40V$; $T_{case} = 0^{\circ}C \text{ to } +70^{\circ}C$; $C_L = 0 \text{ pF}$

Symbol	Parameter	Min	Max	Unit	Figure	Notes
	Frequency	33.33	66.66	MHz		1x CLK
t_1	CLK Period	15		ns	3.1	
t_{1a}	CLK Period Stability		± 250	ps		(18), (19), (20), (21)
t_2	CLK High Time	4		ns	3.1	@2V, (1)
t_3	CLK Low Time	4		ns	3.1	@0.8V, (1)
t_4	CLK Fall Time	0.15	1.5	ns	3.1	(2.0V–0.8V), (1)
t_5	CLK Rise Time	0.15	1.5	ns	3.1	(0.8V–2.0V), (1)
t_6	ADS#, A3–A31, BT0–3, PWT, PCD, BE0–7#, M/IO#, D/C#, W/R#, CACHE#, SCYC, LOCK# Valid Delay	1.5	8.0	ns	3.2	
t_{6a}	AP Valid Delay	1.5	9.5	ns	3.2	
t_7	ADS#, AP, A3–A31, BT0–3, PWT, PCD, BE0–7#, M/IO#, D/C#, W/R#, CACHE#, SCYC, LOCK# Float Delay		10	ns	3.3	(1)
t_8	PCHK#, APCHK#, IERR#, FERR# Valid Delay	1.5	8.3	ns	3.2	(4)
t_9	BREQ, HLDA, SMIACK# Valid Delay	1.5	8.0	ns	3.2	(4)
t_{10}	HIT#, HITM# Valid Delay	1.5	8.0	ns	3.2	
t_{11}	PM0–1, BP0–3, IU, IV, IBT Valid Delay	1.5	10	ns	3.2	
t_{11a}	PRDY Valid Delay	1.5	8.0	ns	3.2	
t_{12}	D0–D63, DP0–7 Write Data Valid Delay	1.5	9	ns	3.2	
t_{13}	D0–63, DP0–7 Write Data Float Delay		10	ns	3.3	(1)
t_{14}	A5–A31 Setup Time	6.5		ns	3.4	
t_{15}	A5–A31 Hold Time	1.5		ns	3.4	
t_{16}	EADS#, INV, AP Setup Time	5		ns	3.4	
t_{17}	EADS#, INV, AP Hold Time	1.5		ns	3.4	
t_{18}	KEN#, WB/WT# Setup Time	5		ns	3.4	
t_{18a}	NA# Setup Time	4.5		ns	3.4	
t_{19}	KEN#, WB/WT#, NA# Hold Time	1.5		ns	3.4	
t_{20}	BRDY# Setup Time	5		ns	3.4	
t_{21}	BRDY# Hold Time	1.5		ns	3.4	
t_{22}	AHOLD, BOFF# Setup Time	5.5		ns	3.4	
t_{23}	AHOLD, BOFF# Hold Time	1.5		ns	3.4	

Table 3-3. 66 MHz Pentium™ Processor A.C. Specifications

 $V_{CC} = 4.9V$ to $5.40V$; $T_{case} = 0^{\circ}C$ to $+70^{\circ}C$; $C_L = 0$ pF (Continued)

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t ₂₄	BUSCHK #, EWBE #, HOLD, PEN # Setup Time	5		ns	3.4	
t ₂₅	BUSCHK #, EWBE #, HOLD, PEN # Hold Time	1.5		ns	3.4	
t ₂₆	A20M #, INTR, Setup Time	5		ns	3.4	(12), (16)
t ₂₇	A20M #, INTR, Hold Time	1.5		ns	3.4	(13)
t ₂₈	INIT, FLUSH #, NMI, SMI #, IGNNE # Setup Time	5		ns	3.4	(16), (17)
t ₂₉	INIT, FLUSH #, NMI, SMI #, IGNNE # Hold Time	1.5		ns	3.4	
t ₃₀	INIT, FLUSH #, NMI, SMI #, IGNNE # Pulse Width, Async	2		CLKs		(15), (17)
t ₃₁	R/S # Setup Time	5		ns	3.4	(12), (16), (17)
t ₃₂	R/S # Hold Time	1.5		ns	3.4	(13)
t ₃₃	R/S # Pulse Width, Async.	2		CLKs		(15), (17)
t ₃₄	D0–D63 Read Data Setup Time	3.8		ns	3.4	
t _{34a}	DP0–7 Read Data Setup Time	3.8		ns	3.4	
t ₃₅	D0–D63, DP0–7 Read Data Hold Time	2		ns	3.4	
t ₃₆	RESET Setup Time	5		ns	3.5	(11), (12), (16)
t ₃₇	RESET Hold Time	1.5		ns	3.5	(11), (13)
t ₃₈	RESET Pulse Width, V_{CC} & CLK Stable	15		CLKs	3.5	(11)
t ₃₉	RESET Active After V_{CC} & CLK Stable	1		ms	3.5	power up, (11)
t ₄₀	Pentium processor Reset Configuration Signals (INIT, FLUSH #, FRCMC #) Setup Time	5		ns	3.5	(12), (16), (17)
t ₄₁	Pentium processor Reset Configuration Signals (INIT, FLUSH #, FRCMC #) Hold Time	1.5		ns	3.5	(13)
t ₄₂	Pentium processor Reset Configuration Signals (INIT, FLUSH #, FRCMC #) Setup Time, Async.	2		CLKs	3.5	(16)
t ₄₃	Pentium processor Reset Configuration Signals (INIT, FLUSH #, FRCMC #) Hold Time, Async.	2		CLKs	3.5	
t ₄₄	TCK Frequency		16	MHz		
t ₄₅	TCK Period	62.5		ns	3.1	

Table 3-3. 66 MHz Pentium™ Processor A.C. Specifications
 $V_{CC} = 4.9V$ to $5.40V$; $T_{case} = 0^{\circ}C$ to $+70^{\circ}C$; $C_L = 0$ pF (Continued)

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{46}	TCK High Time	25		ns	3.1	@2V, (1)
t_{47}	TCK Low Time	25		ns	3.1	@0.8V, (1)
t_{48}	TCK Fall Time		5	ns	3.1	(2.0V–0.8V), (1), (8), (9)
t_{49}	TCK Rise Time		5	ns	3.1	(0.8V–2.0V), (1), (8), (9)
t_{50}	TRST# Pulse Width	40		ns	3.7	Asynchronous, (1)
t_{51}	TDI, TMS Setup Time	5		ns	3.6	(7)
t_{52}	TDI, TMS Hold Time	13		ns	3.6	(7)
t_{53}	TDO Valid Delay	3	20	ns	3.6	(8)
t_{54}	TDO Float Delay		25	ns	3.6	(1), (8)
t_{55}	All Non-Test Outputs Valid Delay	3	20	ns	3.6	(3), (8), (10)
t_{56}	All Non-Test Outputs Float Delay		25	ns	3.6	(1), (3), (8), (10)
t_{57}	All Non-Test Inputs Setup Time	5		ns	3.6	(3), (7), (10)
t_{58}	All Non-Test Inputs Hold Time	13		ns	3.6	(3), (7), (10)

NOTES:

- Not 100% tested. Guaranteed by design/characterization.
- TTL input test waveforms are assumed to be 0 to 3 Volt transitions with 1V/ns rise and fall times.
- Non-Test Outputs and Inputs are the normal output or input signals (besides TCK, TRST#, TDI, TDO, and TMS). These timings correspond to the response of these signals due to boundary scan operations.
- APCHK#, FERR#, HLDA, IERR#, LOCK#, and PCHK# are glitch free outputs. Glitch free signals monotonically transition without false transitions (i.e. glitches).
- $0.8\text{ V/ns} \leq \text{CLK input rise/fall time} \leq 8\text{ V/ns}$.
- $0.3\text{ V/ns} \leq \text{Input rise/fall time} \leq 5\text{ V/ns}$.
- Referenced to TCK rising edge.
- Referenced to TCK falling edge.
- 1ns can be added to the maximum TCK rise and fall times for every 10 MHz of frequency below 16 MHz.
- During probe mode operation, use the normal specified timings. Do not use the boundary scan timings (t_{55-58}).
- FRCMC# should be tied to V_{CC} (high) to ensure proper operation of the Pentium processor as a master Pentium processor.
- Setup time is required to guarantee recognition on a specific clock.
- Hold time is required to guarantee recognition on a specific clock.
- All TTL timings are referenced from 1.5 V.
- To guarantee proper asynchronous recognition, the signal must have been deasserted (inactive) for a minimum of 2 clocks before being returned active and must meet the minimum pulse width.
- This input may be driven asynchronously.
- When driven asynchronously, NMI, FLUSH#, R/S#, INIT, and SMI# must be deasserted (inactive) for a minimum of 2 clocks before being returned active.
- Functionality is guaranteed by design/characterization.
- Measured on rising edge of adjacent CLKs at 1.5V.
- To ensure a 1:1 relationship between the amplitude of the input jitter and the internal and external clocks, the jitter frequency spectrum should not have any power spectrum peaking between 500 KHz and $\frac{1}{3}$ of the CLK operating frequency.
- The amount of jitter present must be accounted for as a component of CLK skew between devices.

Table 3-4. 60 MHz Pentium™ Processor A.C. Specifications

 $V_{CC} = 5V \pm 5\%$; $T_{case} = 0^{\circ}C$ to $+80^{\circ}C$; $C_L = 0$ pF

Symbol	Parameter	Min	Max	Unit	Figure	Notes
	Frequency	33.33	60	MHz		1x CLK
t_1	CLK Period	16.67		ns	3.1	
t_{1a}	CLK Period Stability		± 250	ps		(18), (19), (20), (21)
t_2	CLK High Time	4		ns	3.1	@2V, (1)
t_3	CLK Low Time	4		ns	3.1	@0.8V, (1)
t_4	CLK Fall Time	0.15	1.5	ns	3.1	(2.0V–0.8V), (1)
t_5	CLK Rise Time	0.15	1.5	ns	3.1	(0.8V–2.0V), (1)
t_6	ADS#, A3–A31, BT0–3, PWT, PCD, BE0–7#, M/IO#, D/C#, W/R#, CACHE#, SCYC, LOCK# Valid Delay	1.5	9.0	ns	3.2	
t_{6a}	AP Valid Delay	1.5	10.5	ns	3.2	
t_7	ADS#, AP, A3–A31, BT0–3, PWT, PCD, BE0–7#, M/IO#, D/C#, W/R#, CACHE#, SCYC, LOCK# Float Delay		11	ns	3.3	(1)
t_8	PCHK#, APCHK#, IERR#, FERR# Valid Delay	1.5	9.3	ns	3.2	(4)
t_9	BREQ, HLDA, SMIACK# Valid Delay	1.5	9.0	ns	3.2	(4)
t_{10}	HIT#, HITM# Valid Delay	1.5	9.0	ns	3.2	
t_{11}	PM0–1, BP0–3, IU, IV, IBT Valid Delay	1.5	11	ns	3.2	
t_{11a}	PRDY Valid Delay	1.5	9.0	ns	3.2	
t_{12}	D0–D63, DP0–7 Write Data Valid Delay	1.5	10	ns	3.2	
t_{13}	D0–D63, DP0–7 Write Data Float Delay		11	ns	3.3	(1)
t_{14}	A5–A31 Setup Time	7		ns	3.4	
t_{15}	A5–A31 Hold Time	1.5		ns	3.4	
t_{16}	EADS#, INV, AP Setup Time	5.5		ns	3.4	
t_{17}	EADS#, INV, AP Hold Time	1.5		ns	3.4	
t_{18}	KEN#, WB/WT# Setup Time	5.5		ns	3.4	
t_{18a}	NA# Setup Time	5.0		ns	3.4	
t_{19}	KEN#, WB/WT#, NA# Hold Time	1.5		ns	3.4	
t_{20}	BRDY# Setup Time	5.5		ns	3.4	
t_{21}	BRDY# Hold Time	1.5		ns	3.4	
t_{22}	AHOLD, BOFF# Setup Time	6		ns	3.4	
t_{23}	AHOLD, BOFF# Hold Time	1.5		ns	3.4	
t_{24}	BUSCHK#, EWBE#, HOLD, PEN# Setup Time	5.5		ns	3.4	
t_{25}	BUSCHK#, EWBE#, HOLD, PEN# Hold Time	1.5		ns	3.4	

Table 3-4. 60 MHz Pentium™ Processor A.C. Specifications
 $V_{CC} = 5V \pm 5\%$; $T_{CASE} = 0^{\circ}C$ to $+80^{\circ}C$; $C_L = 0$ pF (Continued)

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t ₂₆	A20M#, INTR, Setup Time	5.5		ns	3.4	(12), (16)
t ₂₇	A20M#, INTR, Hold Time	1.5		ns	3.4	(13)
t ₂₈	INIT, FLUSH#, NMI, SMI#, IGNNE# Setup Time	5.5		ns	3.4	(16), (17)
t ₂₉	INIT, FLUSH#, NMI, SMI#, IGNNE# Hold Time	1.5		ns	3.4	
t ₃₀	INIT, FLUSH#, NMI, SMI#, IGNNE# Pulse Width, Async	2		CLKs		(15), (17)
t ₃₁	R/S# Setup Time	5.5		ns	3.4	(12), (16), (17)
t ₃₂	R/S# Hold Time	1.5		ns	3.4	(13)
t ₃₃	R/S# Pulse Width, Async.	2		CLKs		(15), (17)
t ₃₄	D0–D63 Read Data Setup Time	4.3		ns	3.4	
t _{34a}	DP0–7 Read Data Setup Time	4.3		ns	3.4	
t ₃₅	D0–D63, DP0–7 Read Data Hold Time	2		ns	3.4	
t ₃₆	RESET Setup Time	5.5		ns	3.5	(11), (12), (16)
t ₃₇	RESET Hold Time	1.5		ns	3.5	(11), (13)
t ₃₈	RESET Pulse Width, V _{CC} & CLK Stable	15		CLKs	3.5	(11)
t ₃₉	RESET Active after V _{CC} & CLK Stable	1		ms	3.5	Power Up, (11)
t ₄₀	Pentium Processor Reset Configuration Signals (INIT, FLUSH#, FRCMC#) Setup Time	5.5		ns	3.5	(12), (16), (17)
t ₄₁	Pentium Processor Reset Configuration Signals (INIT, FLUSH#, FRCMC#) Hold Time	1.5		ns	3.5	(13)
t ₄₂	Pentium Processor Reset Configuration Signals (INIT, FLUSH#, FRCMC#) Setup Time, Async.	2		CLKs	3.5	(16)
t ₄₃	Pentium Processor Reset Configuration Signals (INIT, FLUSH#, FRCMC#) Hold Time, Async.	2		CLKs	3.5	
t ₄₄	TCK Frequency		16	MHz		
t ₄₅	TCK Period	62.5		ns	3.1	
t ₄₆	TCK High Time	25		ns	3.1	@2V, (1)
t ₄₇	TCK Low Time	25		ns	3.1	@0.8V, (1)
t ₄₈	TCK Fall Time		5	ns	3.1	(2.0V–0.8V), (1), (8), (9)
t ₄₉	TCK Rise Time		5	ns	3.1	(0.8V–2.0V), (1), (8), (9)

Table 3-4. 60 MHz Pentium™ Processor A.C. Specifications
 $V_{CC} = 5V \pm 5\%$; $T_{CASE} = 0^{\circ}C$ to $+80^{\circ}C$; $C_L = 0$ pF (Continued)

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{50}	TRST# Pulse Width	40		ns	3.7	Async, (1)
t_{51}	TDI, TMS Setup Time	5		ns	3.6	(7)
t_{52}	TDI, TMS Hold Time	13		ns	3.6	(7)
t_{53}	TDO Valid Delay	3	20	ns	3.6	(8)
t_{54}	TDO Float Delay		25	ns	3.6	(1), (8)
t_{55}	All Non-Test Outputs Valid Delay	3	20	ns	3.6	(3), (8), (10)
t_{56}	All Non-Test Outputs Float Delay		25	ns	3.6	(1), (3), (8), (10)
t_{57}	All Non-Test Inputs Setup Time	5		ns	3.6	(3), (7), (10)
t_{58}	All Non-Test Inputs Hold Time	13		ns	3.6	(3), (7), (10)

NOTES:

- Not 100% tested. Guaranteed by design/characterization.
- TTL input test waveforms are assumed to be 0 to 3 Volt transitions with 1V/ns rise and fall times.
- Non-Test Outputs and Inputs are the normal output or input signals (besides TCK, TRST#, TDI, TDO, and TMS). These timings correspond to the response of these signals due to boundary scan operations.
- APCHK#, FERR#, HLDA, IERR#, LOCK#, and PCHK# are glitch free outputs. Glitch free signals monotonically transition without false transitions (i.e. glitches).
- $0.8 \text{ V/ns} \leq \text{CLK input rise/fall time} \leq 8 \text{ V/ns}$.
- $0.3 \text{ V/ns} \leq \text{Input rise/fall time} \leq 5 \text{ V/ns}$.
- Referenced to TCK rising edge.
- Referenced to TCK falling edge.
- 1ns can be added to the maximum TCK rise and fall times for every 10 MHz of frequency below 16 MHz.
- During probe mode operation, use the normal specified timings. Do not use the boundary scan timings (t_{55-58}).
- FRCMC# should be tied to V_{CC} (high) to ensure proper operation of the Pentium processor as a master Pentium processor.
- Setup time is required to guarantee recognition on a specific clock.
- Hold time is required to guarantee recognition on a specific clock.
- All TTL timings are referenced from 1.5 V.
- To guarantee proper asynchronous recognition, the signal must have been deasserted (inactive) for a minimum of 2 clocks before being returned active and must meet the minimum pulse width.
- This input may be driven asynchronously.
- When driven asynchronously, NMI, FLUSH#, R/S#, INIT, and SMI# must be deasserted (inactive) for a minimum of 2 clocks before being returned active.
- Functionality is guaranteed by design/characterization.
- Measured on rising edge of adjacent CLKs at 1.5V.
- To ensure a 1:1 relationship between the amplitude of the input jitter and the internal and external clocks, the jitter frequency spectrum should not have any power spectrum peaking between 500 KHz and $\frac{1}{3}$ of the CLK operating frequency.
- The amount of jitter present must be accounted for as a component of CLK skew between devices.

	Min	Max	Unit	Figure	Notes
t_{50}	40		ns	3.7	Async, (1)
t_{51}	5		ns	3.6	(7)
t_{52}	13		ns	3.6	(7)
t_{53}	3	20	ns	3.6	(8)
t_{54}		25	ns	3.6	(1), (8)
t_{55}	3	20	ns	3.6	(3), (8), (10)
t_{56}		25	ns	3.6	(1), (3), (8), (10)
t_{57}	5		ns	3.6	(3), (7), (10)
t_{58}	13		ns	3.6	(3), (7), (10)

Each valid delay is specified for a 0 pF load. The system designer should use I/O buffer modeling to account for signal flight time delays.

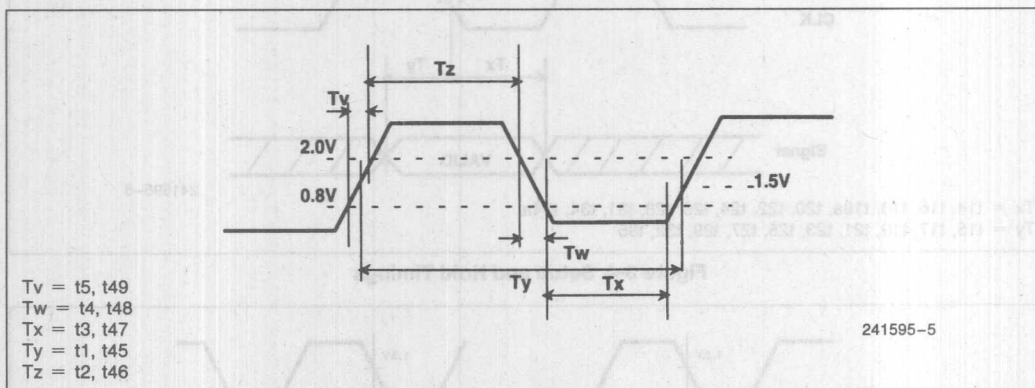


Figure 3-1. Clock Waveform

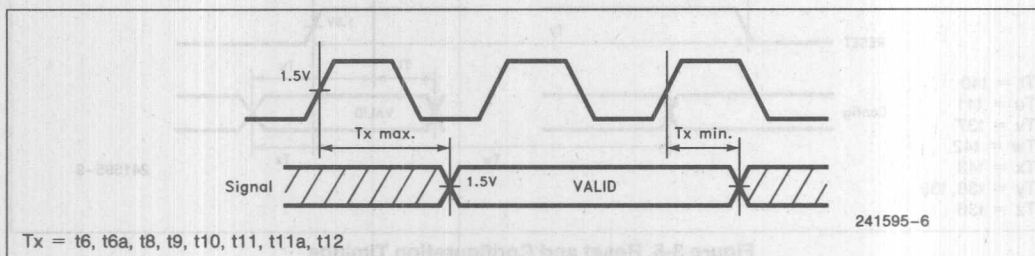


Figure 3-2. Valid Delay Timings

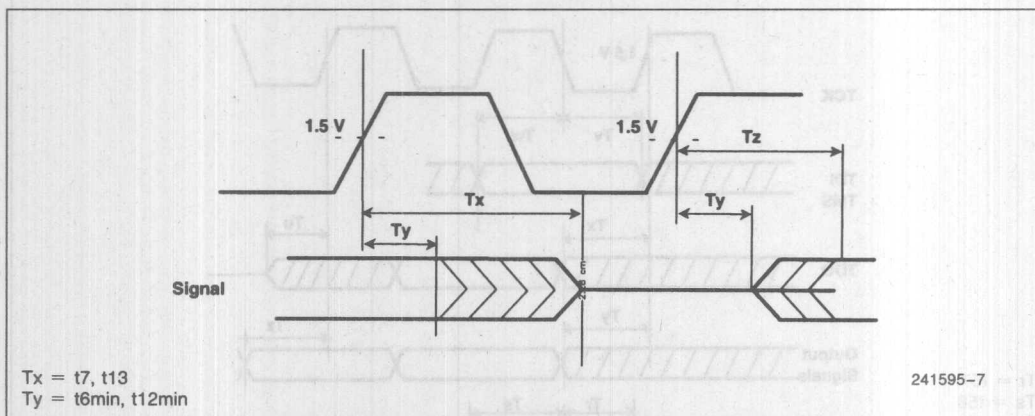


Figure 3-3. Float Delay Timings

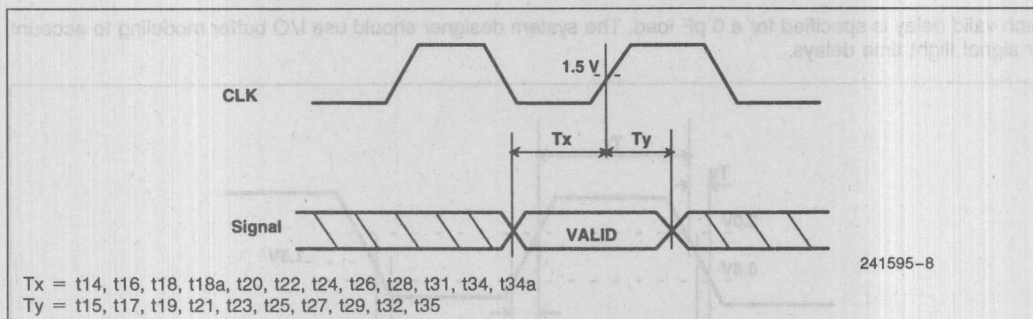


Figure 3-4. Setup and Hold Timings

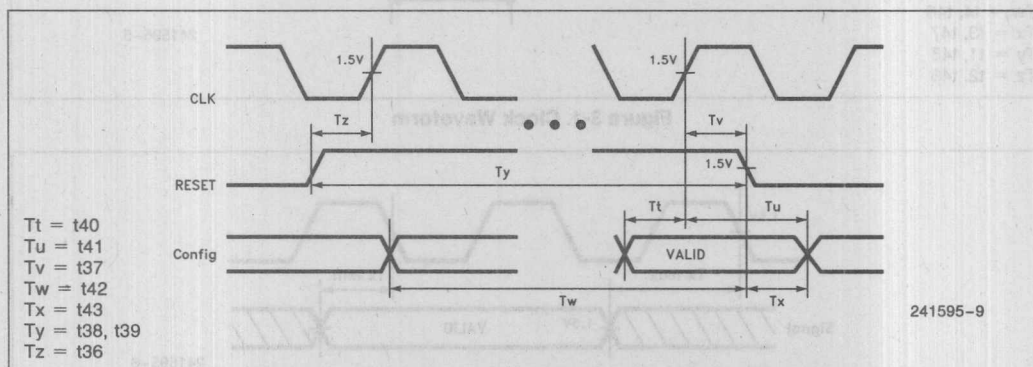


Figure 3-5. Reset and Configuration Timings

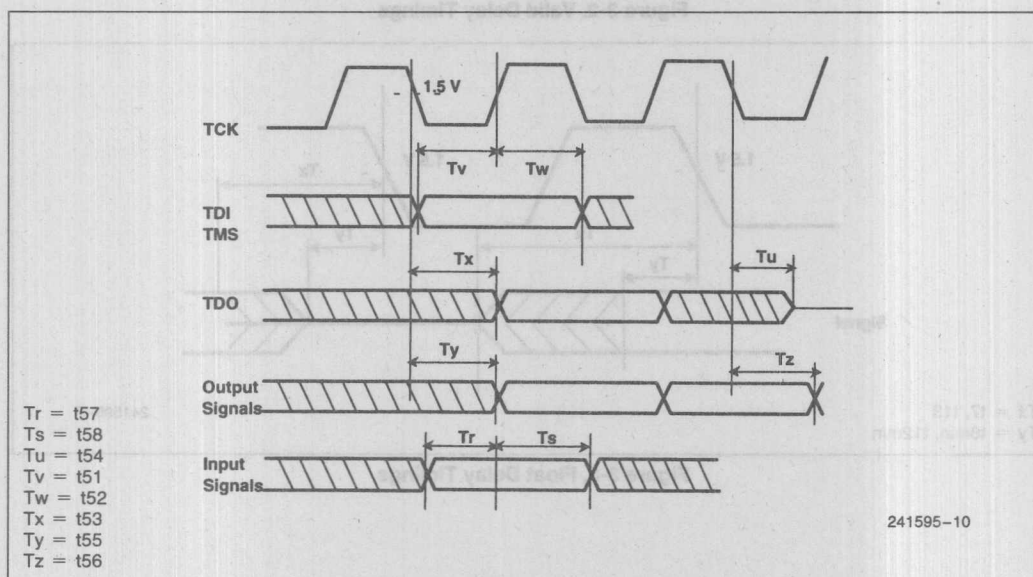


Figure 3-6. Test Timings

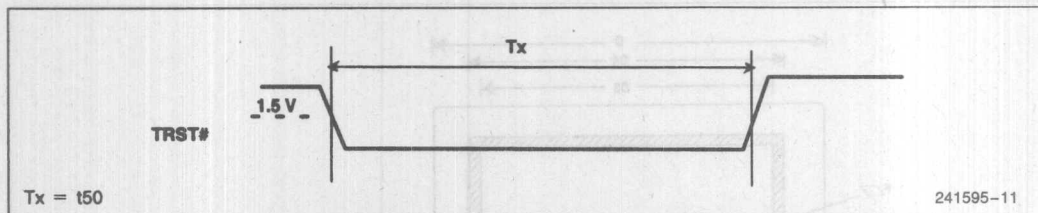


Figure 3-7. Test Reset Timings

4.0 MECHANICAL SPECIFICATIONS

The Pentium processor is packaged in a 273 pin ceramic pin grid array (PGA). The pins are arranged in a 21 by 21 matrix and the package dimensions are 2.16" × 2.16" (Table 4-1).

Figure 4-1 shows the package dimensions for the Pentium processor. The mechanical specifications are provided in Table 4-2.

Table 4-1. Pentium™ Processor Package Information Summary

	Package Type	Total Pins	Pin Array	Package Size	Estimated Wattage
Pentium Processor	PGA	273	21 × 21	2.16" × 2.16" 5.49 cm × 5.49 cm	16

NOTE:

See D.C. Specifications for more detailed power specifications.

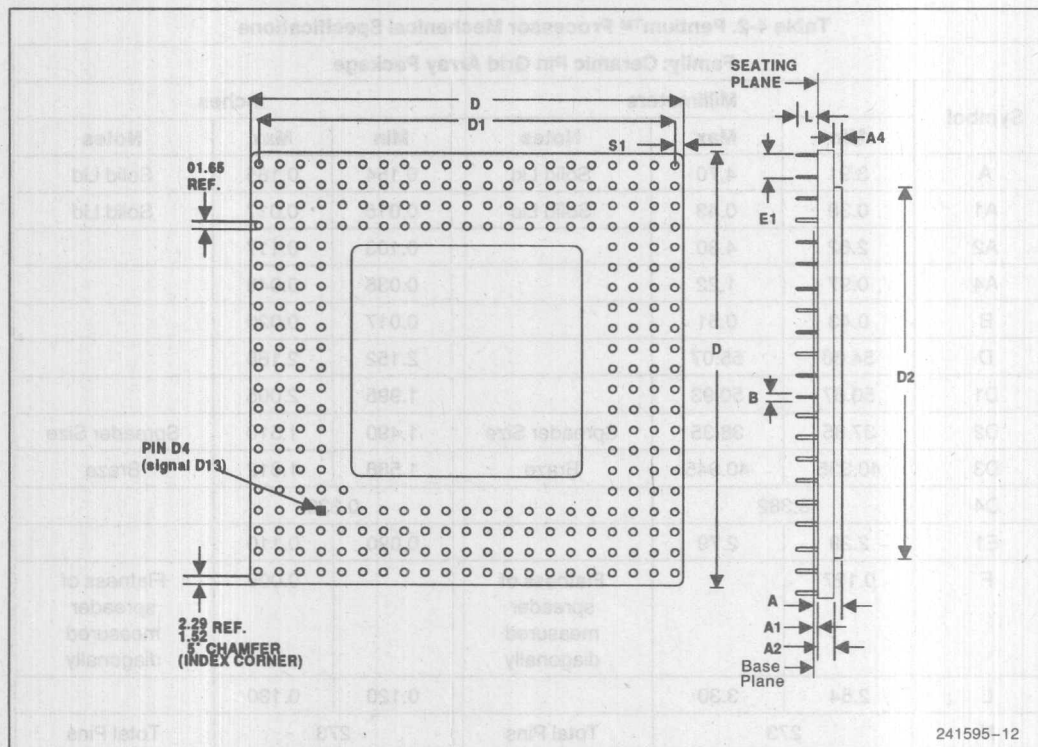


Figure 4-1. Pentium™ Processor Package Dimensions

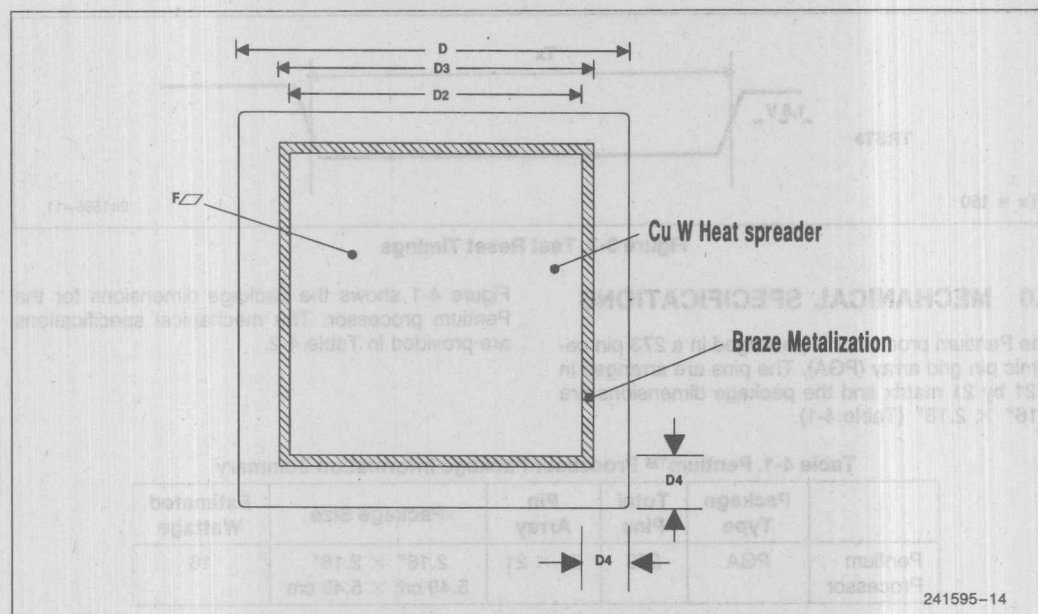


Figure 4-2. Pentium™ Processor Package Dimensions

Table 4-2. Pentium™ Processor Mechanical Specifications

Family: Ceramic Pin Grid Array Package						
Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
A	3.91	4.70	Solid Lid	0.154	0.185	Solid Lid
A1	0.38	0.43	Solid Lid	0.015	0.017	Solid Lid
A2	2.62	4.30		0.103	0.117	
A4	0.97	1.22		0.038	0.048	
B	0.43	0.51		0.017	0.020	
D	54.66	55.07		2.152	2.168	
D1	50.67	50.93		1.995	2.005	
D2	37.85	38.35	Spreader Size	1.490	1.510	Spreader Size
D3	40.335	40.945	Braze	1.588	1.612	Braze
D4	8.382			0.330		
E1	2.29	2.79		0.090	0.110	
F	0.127		Flatness of spreader measured diagonally		0.005	Flatness of spreader measured diagonally
L	2.54	3.30		0.120	0.130	
N	273		Total Pins	273		Total Pins
S1	1.651	2.16		0.065	0.085	

5.0 THERMAL SPECIFICATIONS

The Pentium processor is specified for proper operation when T_C (case temperature) is within the specified range. To verify that the proper T_C is maintained, it should be measured at the center of the top surface (opposite of the pins) of the device in question. To minimize the measurement errors, it is recommended to use the following approach:

- Use 36 gauge or finer diameter k, t, or j type thermocouples. The laboratory testing was done using a thermocouple made by Omega (part number: 5TC-TTK-36-36).

- Attach the thermocouple bead or junction to the center of the package top surface using high thermal conductivity cements. The laboratory testing was done by using Omega Bond (part number: OB-100).
- The thermocouple should be attached at a 90 degrees angle as shown in Figure 5-1. When a heat sink is attached, a hole (no larger than 0.15") should be drilled through the heat sink to allow probing the center of the package as shown in Figure 5-1.
- If the case temperature is measured with a heat sink attached to the package, drill a hole through the heat sink to route the thermocouple wire out.

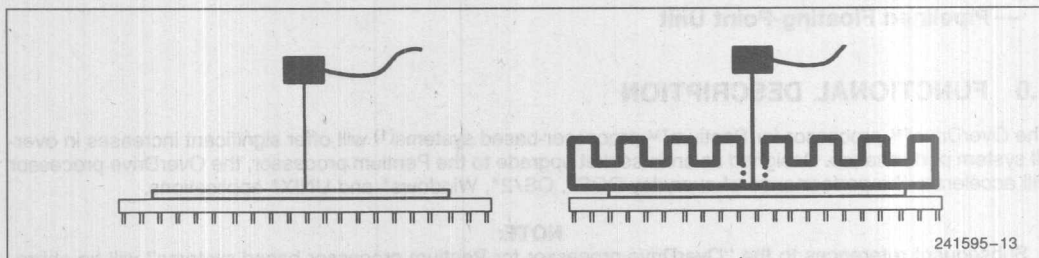


Figure 5-1. Technique for Measuring T_{case}

An ambient temperature T_A is not specified directly. The only restriction is that T_C is met. To determine the allowable T_A values, the following equations may be used:

$$T_J = T_C + (P \cdot \theta_{JC})$$

$$T_A = T_J - (P \cdot \theta_{JA})$$

$$\theta_{CA} = \theta_{JA} - \theta_{JC}$$

$$T_A = T_C - (P \cdot \theta_{CA})$$

where, T_J , T_A , and T_C = Junction, Ambient and Case Temperature, respectively.

θ_{JC} , θ_{JA} , and θ_{CA} = Junction-to-Case, Junction-to-Ambient, and Case-to-Ambient Thermal Resistance, respectively.

P = Maximum Power Consumption

Table 5-1 lists the θ_{JC} and θ_{CA} values for the Pentium processor.

Table 5-1. Junction-to-Case and Case-to-Ambient Thermal Resistances for the Pentium™ Processor (With and Without a Heat Sink)

	θ_{JC}	θ_{CA} vs Airflow (ft/min)					
		0	200	400	600	800	1000
With 0.25" Heat Sink	0.6	8.3	5.4	3.5	2.6	2.1	1.8
With 0.35" Heat Sink	0.6	7.4	4.5	3.0	2.2	1.8	1.6
With 0.65" Heat Sink	0.6	5.9	3.0	1.9	1.5	1.2	1.1
Without Heat Sink	1.2	10.5	7.9	5.5	3.8	2.8	2.4

NOTE:

1. Heat Sink: 2.1 sq. in. base, omni-directional pin Al heat sink with 0.050 in. pin width, 0.143 in pin-to-pin center spacing and 0.150 in. base thickness. Heat sinks are attached to the package with a 2 to 4 mil thick layer of typical thermal grease. The thermal conductivity of this grease is about 1.2 w/m °C.

2. θ_{CA} values shown in Table 5-1. are typical values. The actual θ_{CA} values depend on the air flow in the system (which is typically unsteady, non uniform and turbulent) and thermal interactions between Pentium™ processor and surrounding components though PCB and the ambient.

OverDrive™ PROCESSOR FOR PENTIUM™ PROCESSOR-BASED SYSTEMS SOCKET SPECIFICATION

- **Powerful Performance Boosters for Pentium™ Processor-Based Systems**
 - Improve Overall System Performance by an average of 50%
 - Increase Both Integer and Floating-Point Performance
- **Superscalar Architecture**
 - Two Pipelined Integer Units
 - Capable of under One Clock per Instruction
 - Pipelined Floating-Point Unit
- **OverDrive Processors Upgrade Systems Based on**
 - 60 MHz Pentium Processors
 - 66 MHz Pentium Processors
- **Binary Compatible with Large Installed Software Base**
 - MS-DOS*, OS/2*, Windows*
 - UNIX* System V/386
 - iRMX®, iRMK Kernels

1.0 FUNCTIONAL DESCRIPTION

The OverDrive™ processor for Pentium™ processor-based systems⁽¹⁾ will offer significant increases in overall system performance. Designed as an in-socket upgrade to the Pentium processor, the OverDrive processor will accelerate the performance of everyday DOS*, OS/2*, Windows* and UNIX* applications.

NOTE:

1. Subsequent references to the "OverDrive processor for Pentium processor-based systems" will be abbreviated to "OverDrive processor" in this document. The OverDrive processor for Pentium processor-based systems will be referred to as the "Higher performance Pentium OverDrive processor" in end-user advertising during the 1993-1994 timeframe.

The OverDrive processor will be socket compatible with the Pentium processor. The OverDrive processor for Pentium processor-based systems has the same pinout and A.C. timing specifications as the Pentium processor. The OverDrive processor is packaged in a 273-pin, ceramic, pin grid array package with an attached fan/heatsink present on the OverDrive processor chip. The active cooling solution will be composed of heat sink with attached fan. See Sections 5 and 6 in this document for OverDrive processor mechanical and thermal information. OEMs should be sure to address the OverDrive processor's additional vertical height and required horizontal clearance due to the active cooling solution.

Performance monitoring, a feature which allows trace instruction execution in order to optimize software code, will not be implemented the same on the Pentium processor and the OverDrive processor for Pentium processor based systems.

With the exception of the OverDrive processor's mechanical, thermal and performance monitoring differences from the Pentium processor, the OverDrive processor's functional characteristics will be compatible with the Pentium processor.

Without Heat Sink	With 0.38" Heat Sink	With 0.58" Heat Sink	With 0.78" Heat Sink	With 0.98" Heat Sink	With 1.18" Heat Sink	With 1.38" Heat Sink
1.3	1.0	0.8	0.6	0.4	0.2	0.1
1.0	0.8	0.6	0.4	0.2	0.1	0.0
0.8	0.6	0.4	0.2	0.1	0.0	0.0
0.6	0.4	0.2	0.1	0.0	0.0	0.0
0.4	0.2	0.1	0.0	0.0	0.0	0.0
0.2	0.1	0.0	0.0	0.0	0.0	0.0
0.1	0.0	0.0	0.0	0.0	0.0	0.0

NOTE: The heat sink is a base, semi-directional pin A heat sink with 0.020 in. pin width, 0.142 in. pin-to-pin center spacing and 0.150 in. pin-to-pin pitch. Heat sink is attached to the package with a 5 to 6 mil thick layer of thermal grease. The thermal conductivity of the grease is about 1.5 W/m°C.

*Other brands and names are the property of their respective owners.

Systems Socket Specification

CONTENTS	PAGE
1.0 FUNCTIONAL DESCRIPTION	2-32
2.0 OverDrive™ PROCESSOR UPGRADABILITY	2-34
2.1 OverDrive™ Processor Upgradability Requirements	2-34
2.2 OverDrive™ Socket	2-34
3.0 PIN DESCRIPTIONS	2-36

CONTENTS	PAGE
4.0 ELECTRICAL SPECIFICATIONS	2-38
4.1 D.C. Specifications	2-38
4.2 A.C. Specifications	2-38
5.0 MECHANICAL SPECIFICATIONS	2-38
6.0 THERMAL SPECIFICATIONS	2-41
7.0 REVISION HISTORY	2-42

2

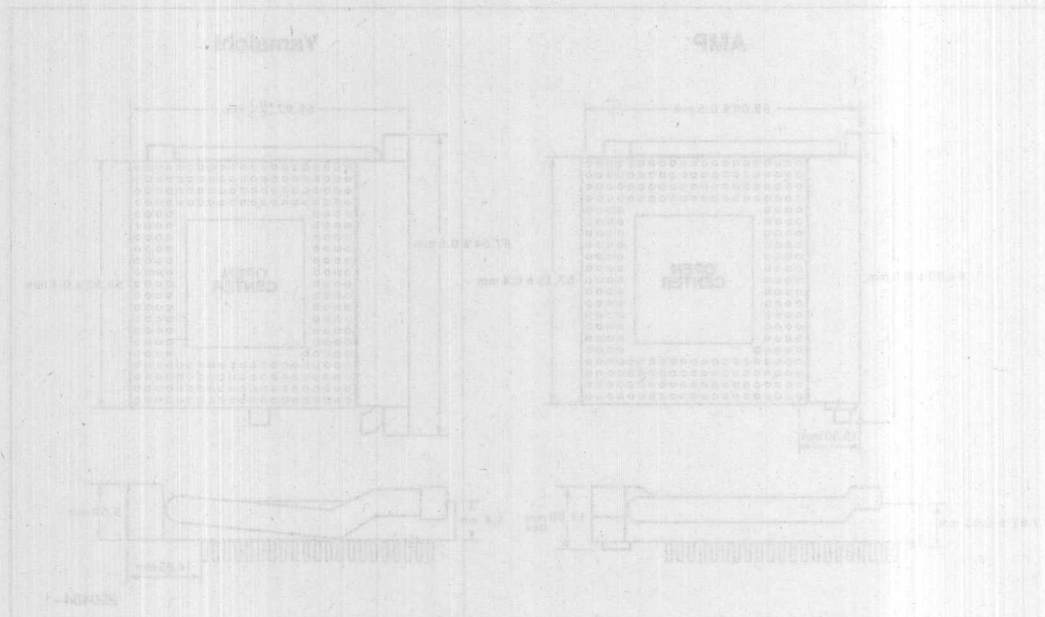


Figure 2-1. OverDrive™ Socket Footprint Dimensions
(See socket manufacturer for the most current information.)

2.0 OverDrive™ PROCESSOR UPGRADABILITY

2.1 OverDrive™ Processor Upgradability Requirements

A Pentium processor based system should be designed to meet certain requirements to support upgradability.

The system must feature a 273-pin, Zero Insertion Force ("ZIF") socket.

The system, as originally shipped, must carry an Intel Pentium processor in the OverDrive socket.

The OverDrive socket should be in a visible and easily accessible location to facilitate end user removal of the Pentium processor and OverDrive processor installation in the same socket. Unacceptable locations or conditions would include placement beneath daughter cards, or which require removal of disk drives or power supplies. Removal of bus cards to permit end user access to the OverDrive socket would be acceptable.

The system should allow clearance of 1.2" above the processor for the OverDrive processor's integrated fan/heatsink. This clearance is divided into the size of the fan/heatsink and the free space

above the fan/heatsink needed to ensure proper air flow.

The thermal requirements are described in Section 6 of this document.

The system should allow full movement of the OverDrive socket handle. For example, any heat sink attached to the Pentium processor should not overhang the processor in any way that would impede full movement of the OverDrive socket handle.

The system must not require any hardware or software modifications to operate the OverDrive processor, including, but not limited to, jumper or switch setting changes, and/or BIOS or logic changes; e.g., jumper changes for frequency selection are acceptable if they are optional and not required for OverDrive processor operation.

OverDrive processor installation in the OEM system must not affect the system warranty.

OEM should provide end user documentation with the OEM system describing the OverDrive processor installation process.

2.2 OverDrive™ Socket

The following drawings in Figure 2-1 show the preliminary worst case OverDrive socket footprints from

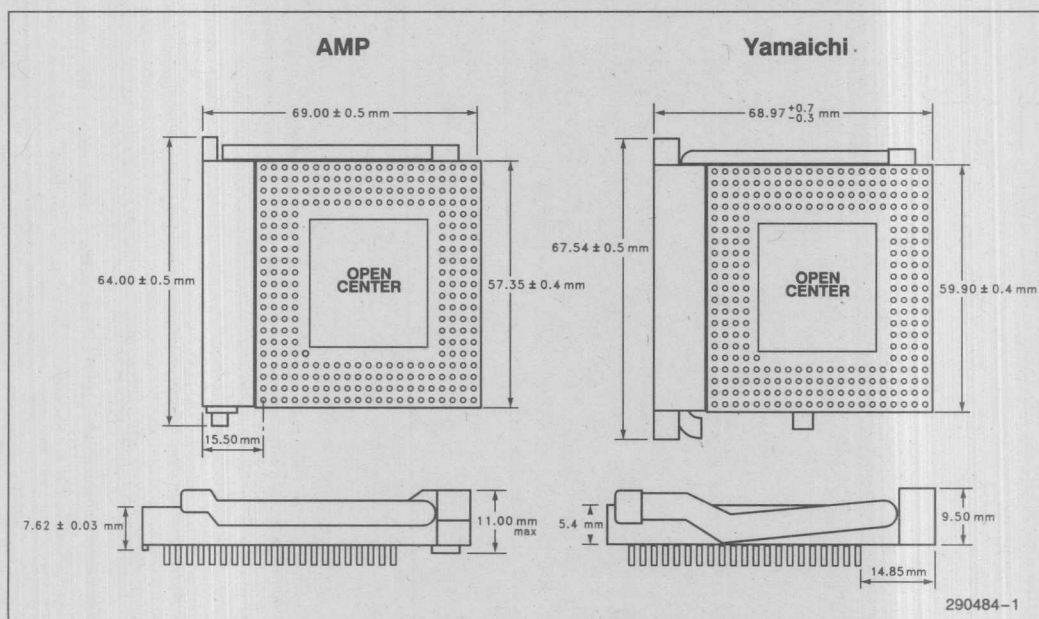


Figure 2-1. OverDrive™ Socket Footprint Dimensions
(See socket manufacturer for the most current information.)

2 potential OverDrive socket vendors, AMP and Yamaichi. OEMs should work directly with socket vendors for the most current socket information.

Figure 2-2 shows the OverDrive processor chip's orientation in the OverDrive socket.

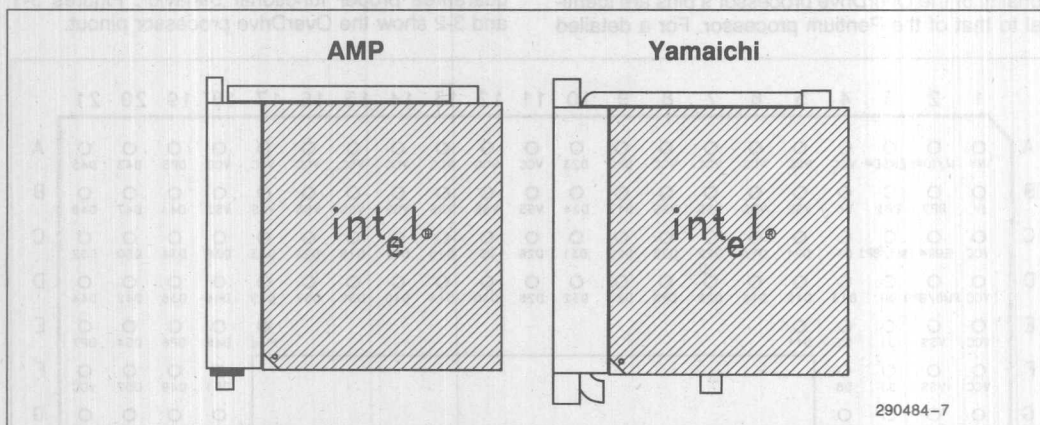


Figure 2-2. Chip Orientation in OverDrive™ Socket

3.0 PIN DESCRIPTIONS

The OverDrive processor pinout as well as the functionality of the OverDrive processor's pins are identical to that of the Pentium processor. For a detailed

description, see the Hardware Interface chapter in the *Pentium Processor Data Book*. Note that all input pins must meet their A.C./D.C. specifications to guarantee proper functional behavior. Figures 3-1 and 3-2 show the OverDrive processor pinout.

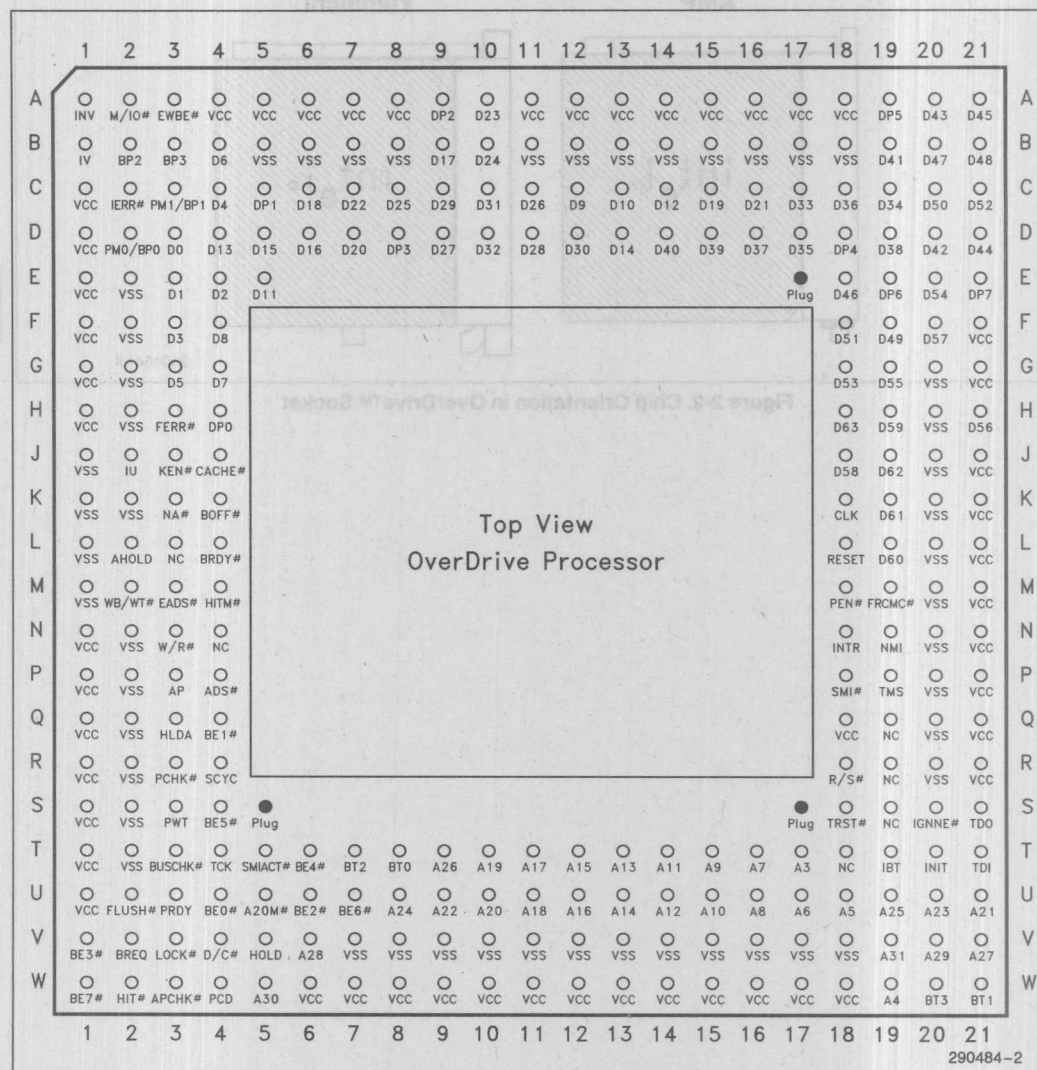


Figure 3-1. OverDrive™ Processor Pinout (Top View)

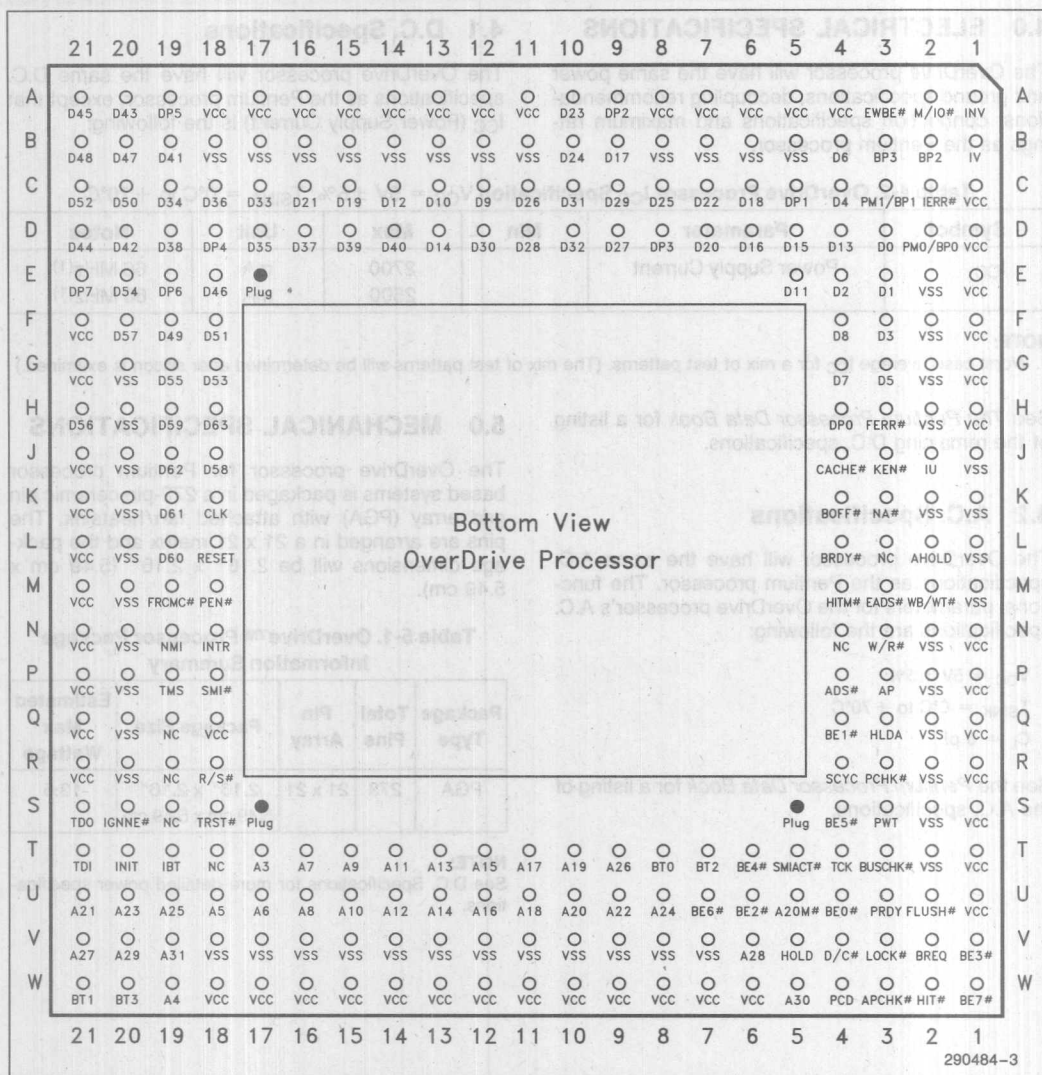


Figure 3-2. OverDrive™ Processor Pinout (Bottom View)

Locations E17, S17 and S5 should be plugged on the Pentium processor/OverDrive processor socket in order to ensure that the Pentium processor/OverDrive processor chip is installed in the socket with the correction orientation.

Locations N4 and L3 are ADSC# and BRDYC# respectively if the 82496 cache controller and 82491 cache SRAMs are used.

4.0 ELECTRICAL SPECIFICATIONS

The OverDrive processor will have the same power and ground specifications, decoupling recommendations, connection specifications and maximum ratings as the Pentium processor.

Table 4-2. OverDrive Processor I_{CC} Specification $V_{CC} = 5V \pm 5\%$, $T_{SINK} = 0^{\circ}C$ to $+70^{\circ}C$

Symbol	Parameter	Min	Max	Unit	Notes
I_{CC}	Power Supply Current		2700 2500	mA mA	66 MHz ⁽¹⁾ 60 MHz ⁽¹⁾

NOTE:

1. Worst case average I_{CC} for a mix of test patterns. (The mix of test patterns will be determined after silicon is examined.)

See *The Pentium Processor Data Book* for a listing of the remaining D.C. specifications.

4.2 A.C. Specifications

The OverDrive processor will have the same A.C. specifications as the Pentium processor. The functional parameters for the OverDrive processor's A.C. specifications are the following:

$$V_{CC} = 5V \pm 5\%$$

$$T_{SINK} = 0^{\circ}C \text{ to } +70^{\circ}C$$

$$C_L = 0 \text{ pF}$$

See the *Pentium Processor Data Book* for a listing of the A.C. specifications.

4.1 D.C. Specifications

The OverDrive processor will have the same D.C. specifications as the Pentium processor, except that I_{CC} (Power Supply Current) is the following:

5.0 MECHANICAL SPECIFICATIONS

The OverDrive processor for Pentium processor based systems is packaged in a 273-pin ceramic pin grid array (PGA) with attached fan/heatsink. The pins are arranged in a 21 x 21 matrix and the package dimensions will be 2.16" x 2.16" (5.49 cm x 5.49 cm).

Table 5-1. OverDrive™ Processor Package Information Summary

Package Type	Total Pins	Pin Array	Package Size	Estimated Max Wattage
PGA	273	21 x 21	2.16" x 2.16" 5.49 cm x 5.49 cm	13.5

NOTE:

See D.C. Specifications for more detailed power specifications.

Table 5-2. OverDrive™ Processor Mechanical Specifications

Family: Ceramic Pin Grid Array Package						
Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
A		33.98	Solid Lid*		1.338	Solid Lid*
A1	2.84	3.50	Solid Lid	0.112	0.138	Solid Lid
A2	0.33	0.43	Solid Lid	0.013	0.017	Solid Lid
A3	2.51	3.07		0.099	0.121	
A4		20.32			0.800	
A5	10.16			0.400		
B	0.43	0.51		0.017	0.020	
D	54.61	55.11		2.150	2.170	
D1	50.67	50.93		1.995	2.005	
E1	2.29	2.79		0.090	0.110	
L	3.05	3.30		0.120	0.130	
N	273			273		
S1	1.65	2.16		0.065	0.085	

2

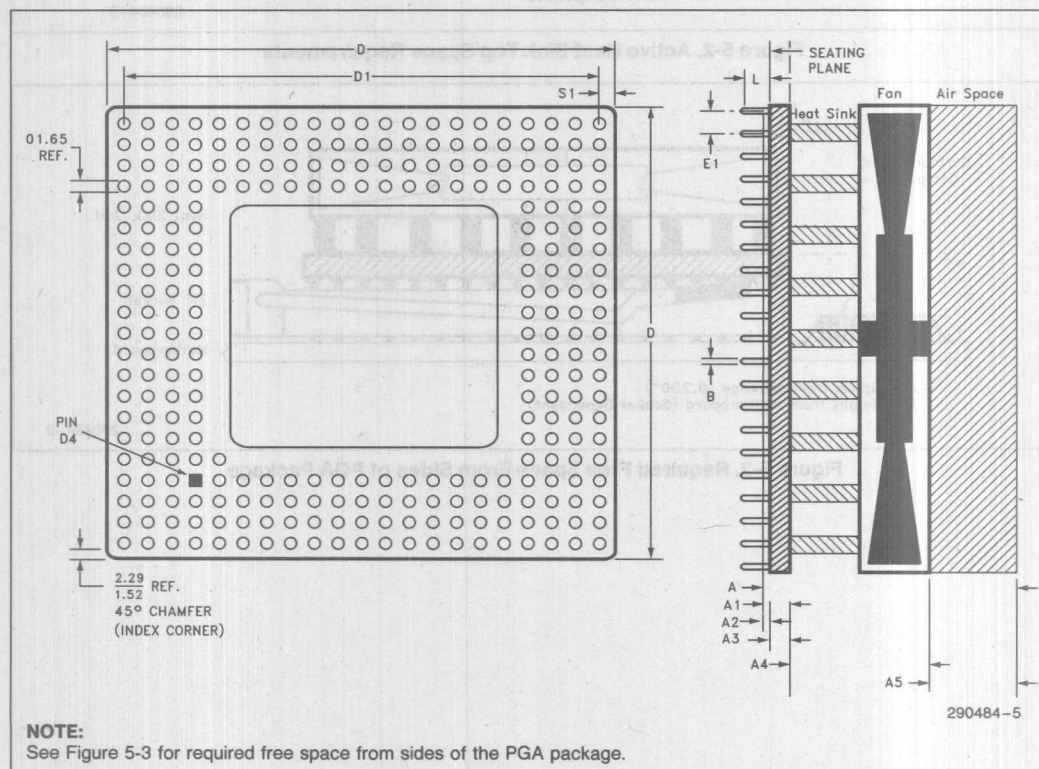


Figure 5-1. OverDrive™ Processor Package Dimensions

As can be seen in the mechanical dimensions in Table 5-2 and Figure 5-1, the actual height required by the heat sink and fan is less than the total space allotted. Since the OverDrive processor for Pentium processor based systems employs a fan/heatsink, a certain amount of space is required above the fan/heatsink unit to ensure that the airflow is not blocked. Figure 5-2 shows unacceptable blocking of the airflow for the OverDrive processor fan/heat-sink. Figure 5-3 details the minimum space needed around the PGA package to ensure proper heat sink airflow.

As shown in Figure 5-3, it is acceptable to allow any device (i.e., add-in cards, surface mount device, chassis, etc.) to enter within the free space distance of 0.2" from the PGA package if it is not taller than the level of the heat sink base. In other words, if a component is taller than height "B", it cannot be closer to the PGA package than distance "A". This applies to all four sides of the PGA package, although the back and handle sides of a ZIF socket will generally automatically meet this specification since they have widths larger than distance "A".

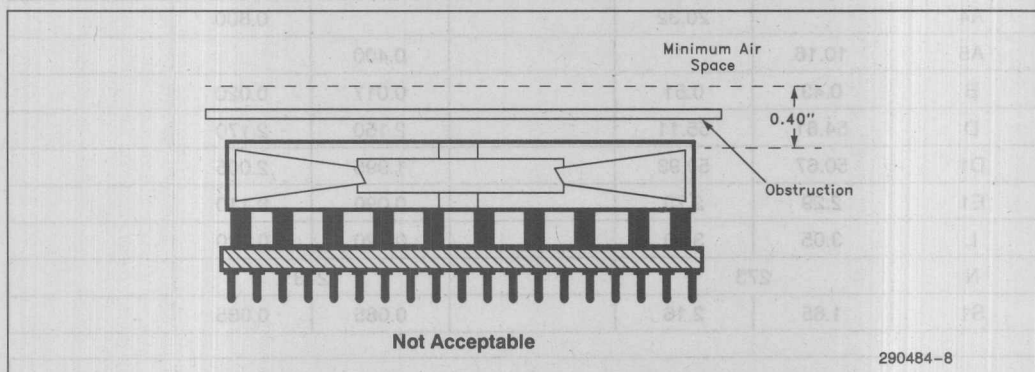


Figure 5-2. Active Heat Sink Top Space Requirements

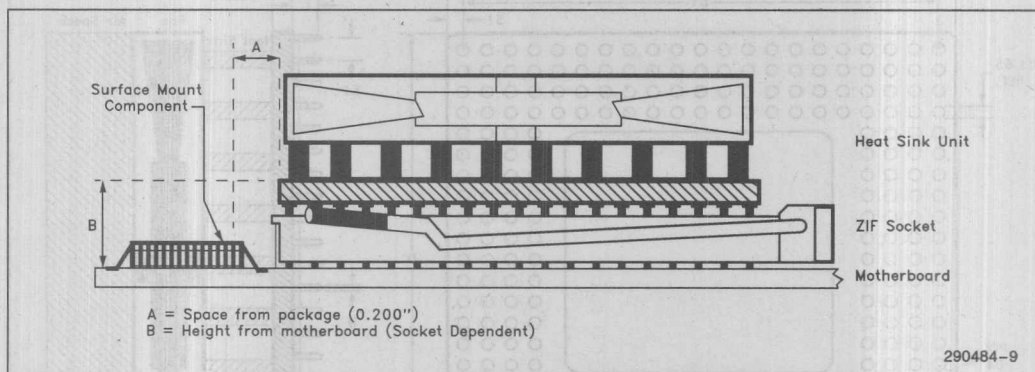


Figure 5-3. Required Free Space From Sides of PGA Package

6.0 THERMAL SPECIFICATIONS

The fan/heatsink cooling solution will properly cool the OverDrive processor as long as the maximum air temperature entering the fan/heatsink cooling solution ($T_{A(in)}$) does not exceed 45°C. It is left up to the OEM to ensure that $T_{A(in)}$ meets this specification by providing sufficient airflow around the OverDrive processor heat sink unit.

Intel's fan/heatsink will dissipate approximately 0.5W and is powered by the chip such that no external wires or connections are required. The extra power needed for the fan is taken into account in the I_{CC} numbers of the processor. Additionally, Intel is evaluating the feasibility of having the OverDrive processor monitor its temperature. No BIOS or hardware changes will be needed for this thermal protection mechanism. The shut down temperature will be greater than the maximum temperature specification of the processor. The fan unit will be designed to be removable so that if fan failure should occur, the unit may be easily replaced. Figure 6-1 gives a functional representation of the processor and heat sink unit. The actual heat sink unit may be different from the one shown in the figure.

Since the OverDrive processor for Pentium processor based systems employs a fan/heatsink, it is not as important that the processor heat sink receive direct airflow, rather that the system has sufficient capability to remove the warm air that the OverDrive processor will generate. This implies that enough airflow exists at the OverDrive processor socket site to keep localized heating from occurring. This can be accomplished by a standard power supply fan with a clear path to the processor. It is recommended that the power supply use a fan that can supply a minimum of **35 CFM** of airflow at the fan exit. This will help ensure that the air exchange rate of the system will be sufficient to meet the OverDrive processor thermal requirements. Figure 6-2 shows how system design can cause localized heating to occur by limiting the airflow in the area of the processor. The airflow supplied in the system should also be enough to ensure that the OEM processor shipped with the system will meet the OEM processor thermal specifications before the system is upgraded with the OverDrive processor.

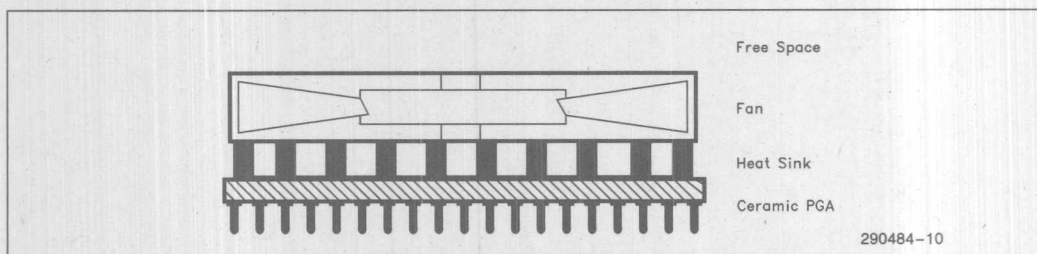
2


Figure 6-1. Active Heat Sink Example

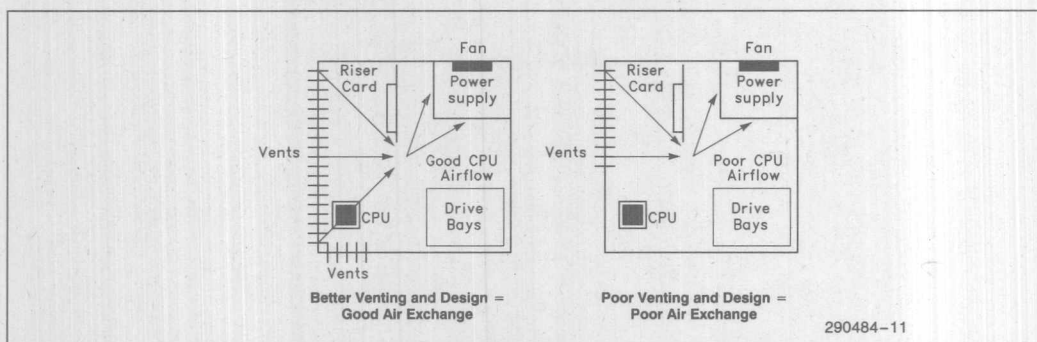


Figure 6-2. OverDrive™ Processor Airflow Design Examples

7.0 REVISION HISTORY

Revision -003 of the *OverDrive™ Processor for Pentium™ Processor-Based Systems Socket Specification* contains updates and improvements to the previous version. A revision summary of major changes is listed below:

Global: Some of the electrical specifications have been removed since they are already shown in the *Pentium Processor Data Book*. These sections now contain references to the *Pentium Processor Data Book*.

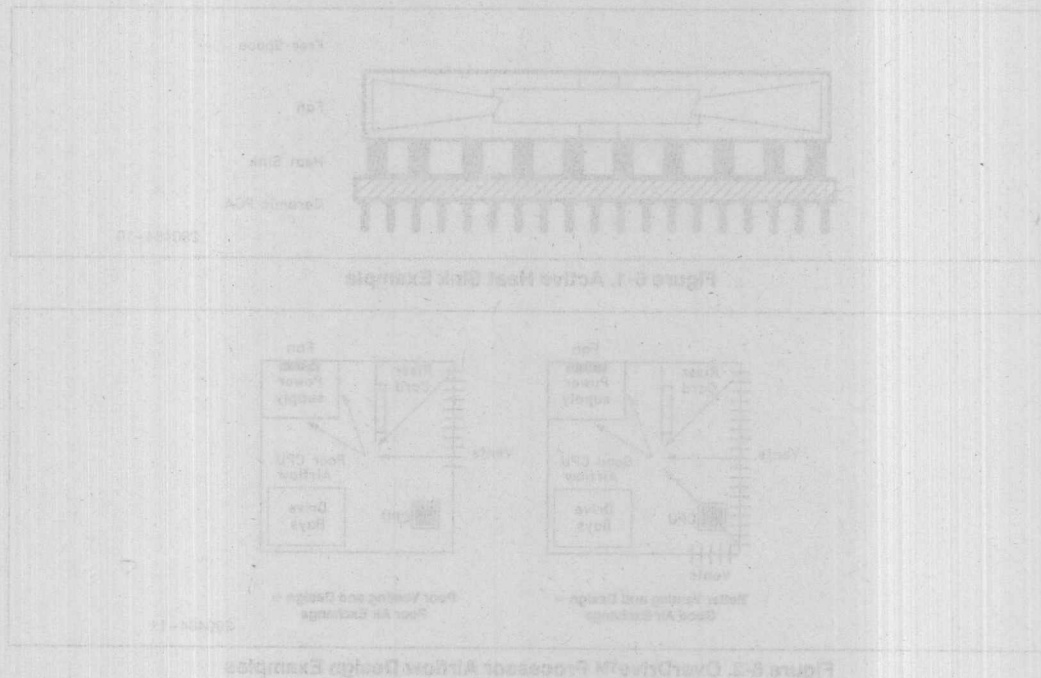
Table 5-2: Table 5-2 contains the following addition errors:

Symbol A1 should have a max tolerance of 3.50mm.

Symbol A should have a max tolerance of 33.98mm.

Symbol A should have a max tolerance of 1.338".

Figure 5-1: Symbol "A" has been corrected to represent the dimension from the package lid to the minimum required free air-space.



For more information regarding the Pentium™ Processor, you may obtain the *Pentium Processor User's Manual* by calling our toll-free literature distribution center at 1-800-548-4725. The *Pentium Processor User's Manual* is a three-volume set with details on Pentium Processor hardware and software design parameters, with specifications also for the 82496 Advanced Cache Controller and 82491 Cache SRAM.

Also listed in this chapter are brief technical profiles of other Pentium Processor peripheral products, including the 82489DX Advanced Interrupt Controller and the 82430 PCIset. For more information on these products, please consult the 1994 Peripherals Handbook.



2

241815-1



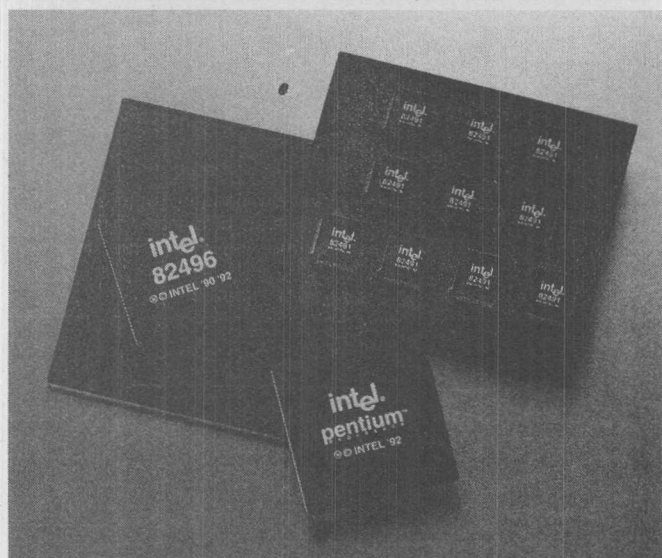
82496 CACHE CONTROLLER AND 82491 CACHE SRAM FOR USE WITH THE Pentium™ PROCESSOR

- **High Performance Second Level Cache**
 - Zero Wait States at 66 MHz
 - Two-way Set Associative
 - Write-Back with MESI Protocol
 - Concurrent CPU Bus and Memory Bus Operation
 - Boundary Scan
- **Pentium Processor**
 - Chip Set Version of Pentium Processor
 - Superscalar Architecture
 - Enhanced Floating Point
 - On-chip 8K Code and 8K Data Caches
 - See Pentium™ Processor User's Manual Volume 2 for more information
- **Highly Flexible**
 - 256K to 512K with parity
 - 32, 64, or 128-Bit Wide Memory Bus
 - Synchronous, Asynchronous, and Strobed Memory Bus Operation
 - Selectable Bus Widths, Line Sizes, Transfers, and Burst Orders
- **Full Multiprocessing Support**
 - Concurrent CPU, Memory Bus, and Snoop Operations
 - Complete MESI Protocol
 - Internal/External Parity Generation/Checking
 - Supports Read-for Ownership, Write-Allocation, and Cache-to-Cache Transfers

The 82496 Cache Controller and multiple 82491 Cache SRAMs combine with the Pentium processor to form a CPU Cache chip set designed for high performance servers and function-rich desktops. The high speed interconnect between the CPU and cache components has been optimized to provide zero-wait state operation. This CPU Cache chip set is fully compatible with existing software, and has new data integrity features for mission critical applications.

The 82496 cache controller implements the MESI write-back protocol for full multiprocessing support. Dual ported buffers and registers allow the 82496 to concurrently handle CPU bus, memory bus, and internal cache operation for maximum performance.

The 82491 is a customized high-performance SRAM that supports 32, 64, and 128-bit wide memory bus widths, 16, 32, and 64 byte line sizes, and optional sectoring. The data path between the CPU bus and memory bus is separated by the 82491, allowing the CPU bus to handshake synchronously, asynchronously, or with a strobed protocol, and allowing concurrent CPU bus and memory bus operations.



241814-1



82430 PCIsset FOR THE Pentium™ PROCESSOR

- Supports the Pentium™ Processor at 60 MHz or 66 MHz
- Interfaces the Host and Standard Buses to the Peripheral Component Interconnect (PCI) Local Bus Operating at 30 MHz or 33 MHz
 - Up to 132 Mbytes/sec Transfer Rate
 - Full Concurrency between CPU Host Bus and PCI Bus Transactions
- Integrated Cache Controller Provided for Optional Second Level Cache
 - 256 Kbyte or 512 Kbyte Cache
 - Write-Back or Write-Through Policy
 - Standard or Burst SRAM
- Integrated Tag RAM for Cost Savings on Second Level Cache
- Provides a 64-Bit Interface to DRAM Memory
 - From 2 Mbytes to 192 Mbytes of Main Memory
 - 70 ns and 60 ns DRAMs Supported
- Supports the Pipelined Address Mode of the Pentium Processor for Higher Performance
- Optional ISA or EISA Standard Bus Interface
 - Single Component ISA Controller
 - Two Component EISA Bus Interface
 - Minimal External Logic Required
- Supports Burst Read and Writes of Memory from the Host and PCI Buses
- Five Integrated Write Posting and Read Prefetch Buffers Increase CPU and PCI Master Performance
- Host CPU Writes to PCI in Zero Wait State PCI Bursts with Optional TRDY # Connection
- Integrated Low Skew Host Bus Clock Driver for Cost and Board Space Savings
- PCIsset Operates Synchronous to the 66 MHz CPU and 33 MHz PCI Clocks
- Byte Parity Support for the Host/PCI and Main Memory Buses
 - Optional Parity on the Second Level Cache

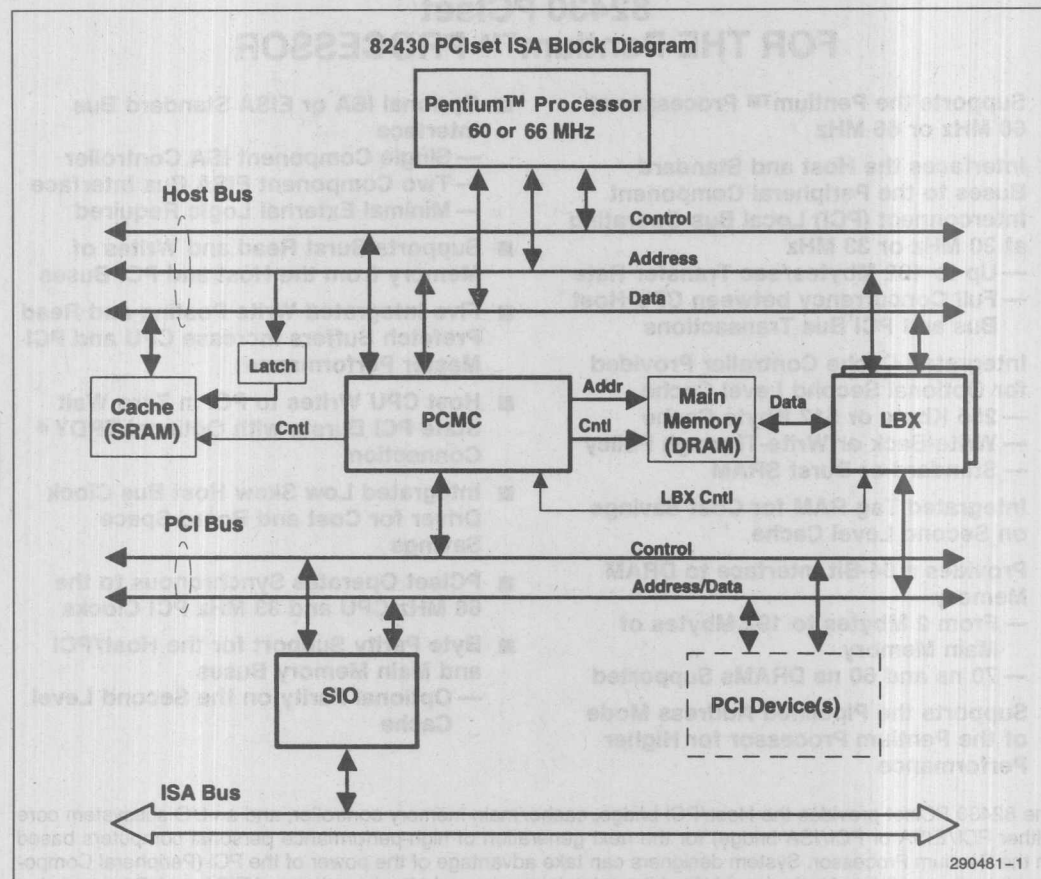
2

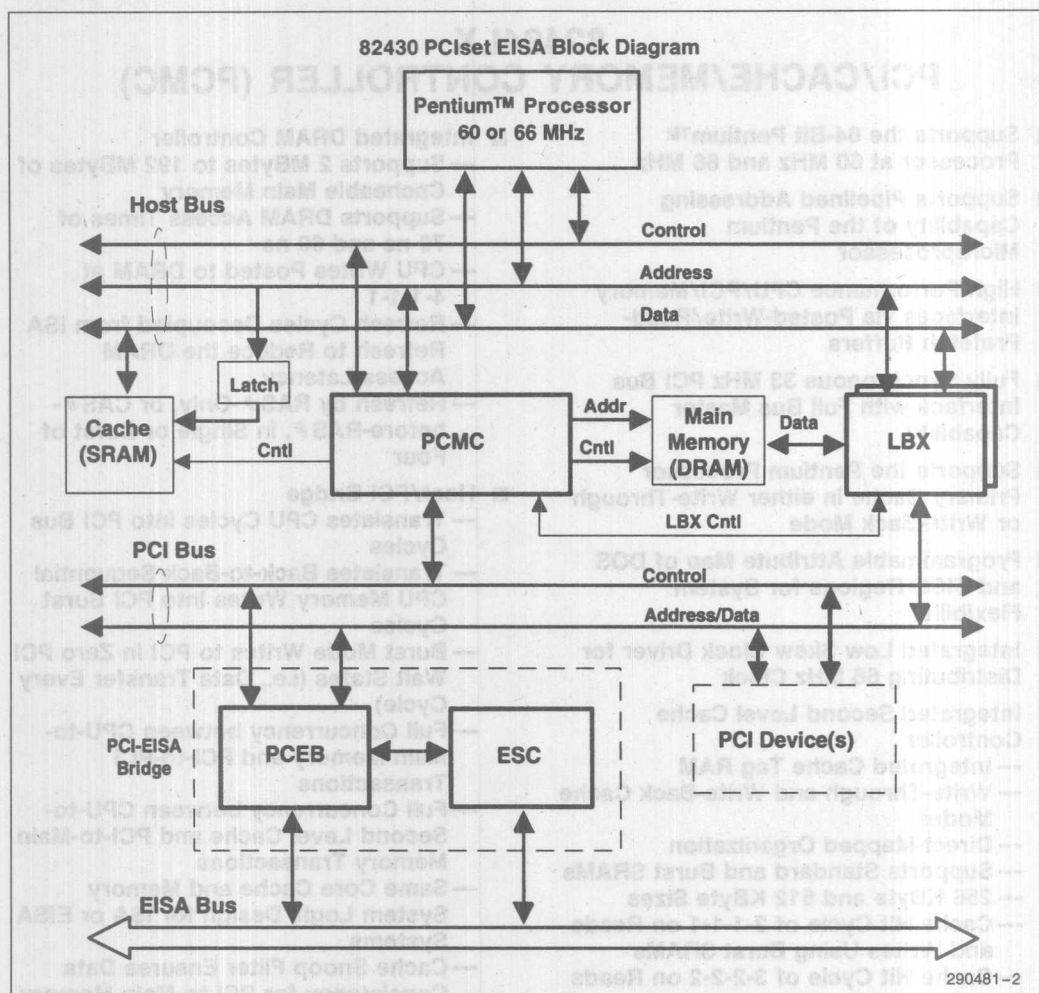
The 82430 PCIsset provides the Host/PCI bridge, cache/main memory controller, and an I/O subsystem core (either PCI/EISA or PCI/ISA bridge) for the next generation of high-performance personal computers based on the Pentium Processor. System designers can take advantage of the power of the PCI (Peripheral Component Interconnect) bus for the local I/O while maintaining access to the large base of EISA and ISA expansion cards, and corresponding software applications. Extensive buffering and buffer management within the bridges ensures maximum efficiency in all three bus environments (Host CPU, PCI, and EISA/ISA Buses).

The 82430 PCIsset consists of the 82434LX PCI/Cache/Memory Controller (PCMC) and the 82433LX Local Bus Accelerator (LBX) components, plus, either a PCI/ISA bridge or a PCI/EISA bridge. The PCMC and LBX provide the core cache and main memory architecture and serve as the Host/PCI bridge. For an ISA-based system, the 82430 PCIsset includes the 82378 System I/O (SIO) component as the PCI/ISA bridge. For an EISA-based system, the 82430 PCIsset includes the 82375EB PCI/EISA Bridge (PCEB) and the 82374EB EISA System Component (ESC). The PCEB and ESC work in tandem to form the complete PCI/EISA bridge. Both the ISA and EISA-based systems are shown on the following pages.

For complete data sheets on all these devices, refer to Order Number 290482 and 290483.

Pentium is a trademark of Intel Corporation.



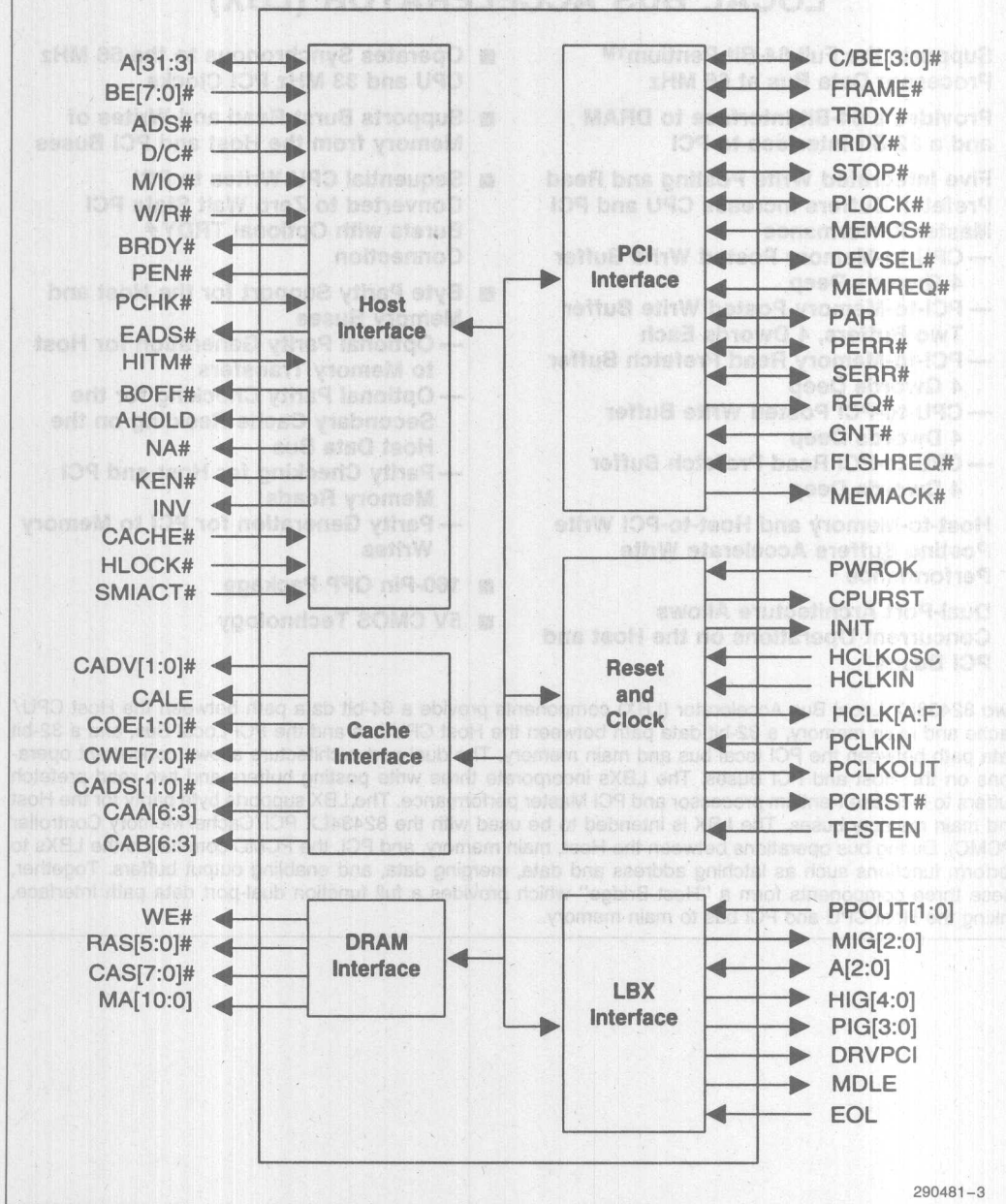


82434LX PCI/CACHE/MEMORY CONTROLLER (PCMC)

- Supports the 64-Bit Pentium™ Processor at 60 MHz and 66 MHz
- Supports Pipelined Addressing Capability of the Pentium Microprocessor
- High Performance CPU/PCI/Memory Interfaces via Posted-Write/Read-Prefetch Buffers
- Fully Synchronous 33 MHz PCI Bus Interface with Full Bus Master Capability
- Supports the Pentium Processor Primary Cache in either Write-Through or Write-Back Mode
- Programmable Attribute Map of DOS and BIOS Regions for System Flexibility
- Integrated Low Skew Clock Driver for Distributing 66 MHz Clock
- Integrated Second Level Cache Controller
 - Integrated Cache Tag RAM
 - Write-Through and Write-Back Cache Modes
 - Direct-Mapped Organization
 - Supports Standard and Burst SRAMs
 - 256 KByte and 512 KByte Sizes
 - Cache Hit Cycle of 3-1-1-1 on Reads and Writes Using Burst SRAMs
 - Cache Hit Cycle of 3-2-2-2 on Reads and 4-2-2-2 on Writes Using Standard SRAMs
- Integrated DRAM Controller
 - Supports 2 MBytes to 192 MBytes of Cacheable Main Memory
 - Supports DRAM Access Times of 70 ns and 60 ns
 - CPU Writes Posted to DRAM at 4-1-1-1
 - Refresh Cycles Decoupled from ISA Refresh to Reduce the DRAM Access Latency
 - Refresh by RAS#-Only, or CAS#-before-RAS#, in Single or Burst of Four
- Host/PCI Bridge
 - Translates CPU Cycles into PCI Bus Cycles
 - Translates Back-to-Back Sequential CPU Memory Writes into PCI Burst Cycles
 - Burst Mode Writes to PCI in Zero PCI Wait States (i.e., Data Transfer Every Cycle)
 - Full Concurrency between CPU-to-Main Memory and PCI-to-PCI Transactions
 - Full Concurrency between CPU-to-Second Level Cache and PCI-to-Main Memory Transactions
 - Same Core Cache and Memory System Logic Design for ISA or EISA Systems
 - Cache Snoop Filter Ensures Data Consistency for PCI-to-Main Memory Transactions
- PCMC (208-Pin QFP Package) Uses 5V CMOS Technology

The 82434LX PCI, Cache, Memory Controller (PCMC) integrates the cache and main memory DRAM control functions and provides the bus control for transfers between the CPU, cache, main memory, and the Peripheral Component Interconnect (PCI) Local Bus. The cache controller supports both write-through and write-back cache policies and cache sizes of 256 KBytes and 512 KBytes. The cache memory can be implemented with either standard or burst SRAMs. The PCMC cache controller integrates a high-performance Tag RAM to reduce system cost. Up to twelve single-sided SIMMs or six double-sided SIMMs provide a maximum of 192 MBytes of main memory. The PCMC is intended to be used with the 82433LX Local Bus Accelerator (LBX). The LBX provides the Host-to-PCI address path and data paths between the CPU/cache, main memory, and PCI. The LBX also contains posted write buffers and read-prefetch buffers. Together, these two components provide a full function data path to main memory and form a PCI bridge to the CPU/Cache and DRAM subsystem.

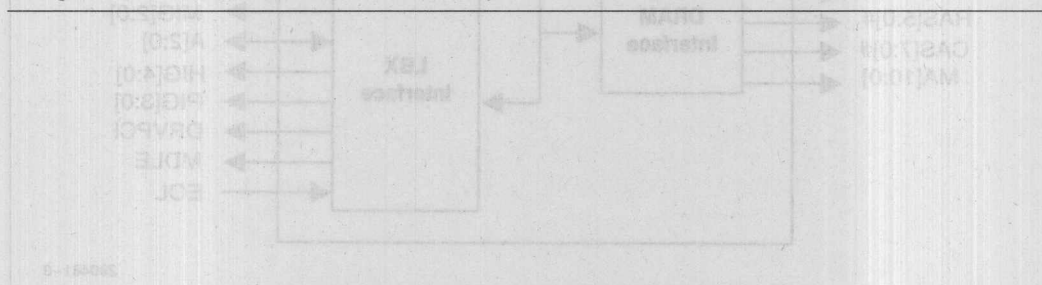
PCMC Block Diagram

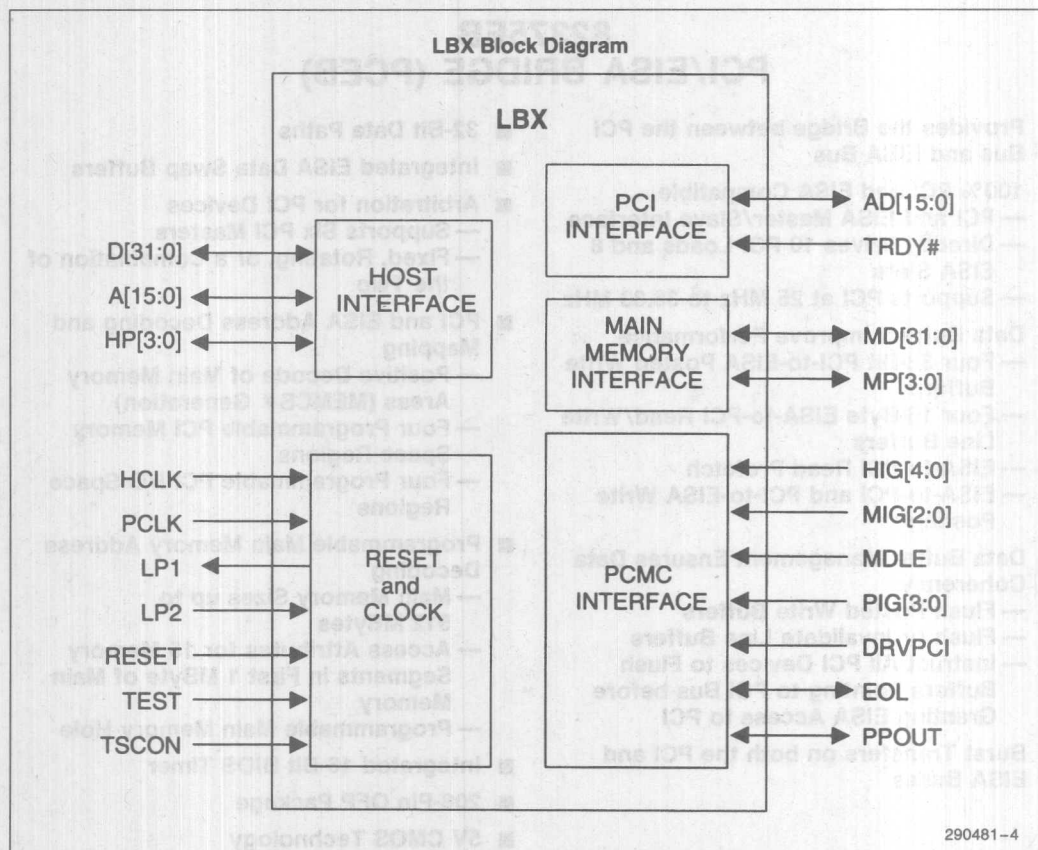


82433LX LOCAL BUS ACCELERATOR (LBX)

- Supports the Full 64-Bit Pentium™ Processor Data Bus at 66 MHz
- Provides a 64-Bit Interface to DRAM and a 32-Bit Interface to PCI
- Five Integrated Write Posting and Read Prefetch Buffers Increase CPU and PCI Master Performance
 - CPU-to-Memory Posted Write Buffer 4 Qwords Deep
 - PCI-to-Memory Posted Write Buffer Two Buffers, 4 Dwords Each
 - PCI-to-Memory Read Prefetch Buffer 4 Qwords Deep
 - CPU-to-PCI Posted Write Buffer 4 Dwords Deep
 - CPU-to-PCI Read Prefetch Buffer 4 Dwords Deep
- Host-to-Memory and Host-to-PCI Write Posting Buffers Accelerate Write Performance
- Dual-Port Architecture Allows Concurrent Operations on the Host and PCI Buses
- Operates Synchronous to the 66 MHz CPU and 33 MHz PCI Clocks
- Supports Burst Read and Writes of Memory from the Host and PCI Buses
- Sequential CPU Writes to PCI Converted to Zero Wait State PCI Bursts with Optional TRDY# Connection
- Byte Parity Support for the Host and Memory Buses
 - Optional Parity Generation for Host to Memory Transfers
 - Optional Parity Checking for the Secondary Cache Residing on the Host Data Bus
 - Parity Checking for Host and PCI Memory Reads
 - Parity Generation for PCI to Memory Writes
- 160-Pin QFP Package
- 5V CMOS Technology

Two 82433LX Local Bus Accelerator (LBX) components provide a 64-bit data path between the Host CPU/cache and main memory, a 32-bit data path between the Host CPU bus and the PCI Local Bus, and a 32-bit data path between the PCI local bus and main memory. The dual-port architecture allows concurrent operations on the Host and PCI Buses. The LBXs incorporate three write posting buffers and two read prefetch buffers to increase Pentium processor and PCI Master performance. The LBX supports byte parity for the Host and main memory buses. The LBX is intended to be used with the 82434LX PCI/Cache/Memory Controller (PCMC). During bus operations between the Host, main memory, and PCI, the PCMC commands the LBXs to perform functions such as latching address and data, merging data, and enabling output buffers. Together, these three components form a "Host Bridge" which provides a full function dual-port data path interface, linking the Host CPU and PCI bus to main memory.

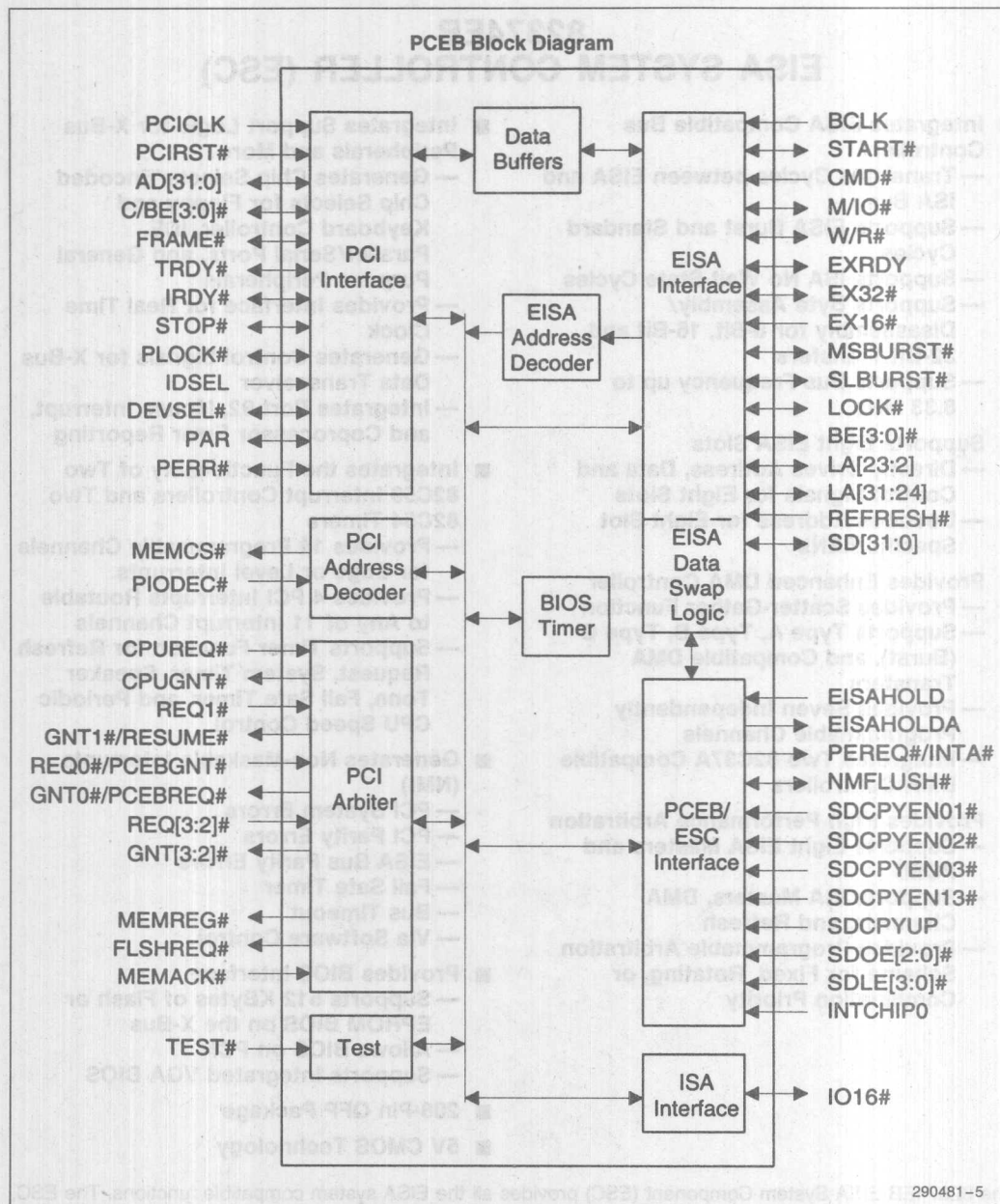




82375EB PCI/EISA BRIDGE (PCEB)

- Provides the Bridge between the PCI Bus and EISA Bus
- 100% PCI and EISA Compatible
 - PCI and EISA Master/Slave Interface
 - Directly Drives 10 PCI Loads and 8 EISA Slots
 - Supports PCI at 25 MHz to 33.33 MHz
- Data Buffers Improve Performance
 - Four 32-Bit PCI-to-EISA Posted Write Buffers
 - Four 16-Byte EISA-to-PCI Read/Write Line Buffers
 - EISA-to-PCI Read Prefetch
 - EISA-to-PCI and PCI-to-EISA Write Posting
- Data Buffer Management Ensures Data Coherency
 - Flush Posted Write Buffers
 - Flush or Invalidate Line Buffers
 - Instruct All PCI Devices to Flush Buffers Pointing to PCI Bus before Granting EISA Access to PCI
- Burst Transfers on both the PCI and EISA Buses
- 32-Bit Data Paths
- Integrated EISA Data Swap Buffers
- Arbitration for PCI Devices
 - Supports Six PCI Masters
 - Fixed, Rotating, or a Combination of the Two
- PCI and EISA Address Decoding and Mapping
 - Positive Decode of Main Memory Areas (MEMCS# Generation)
 - Four Programmable PCI Memory Space Regions
 - Four Programmable PCI I/O Space Regions
- Programmable Main Memory Address Decoding
 - Main Memory Sizes up to 512 MBytes
 - Access Attributes for 15 Memory Segments in First 1 MByte of Main Memory
 - Programmable Main Memory Hole
- Integrated 16-Bit BIOS Timer
- 208-Pin QFP Package
- 5V CMOS Technology

The 82375EB PCI-EISA Bridge (PCEB) provides the master/slave functions on both the Peripheral Component Interconnect (PCI) Local Bus and the EISA Bus. Functioning as a bridge between the PCI and EISA buses, the PCEB provides the address and data paths, bus controls, and bus protocol translation for PCI-to-EISA and EISA-to-PCI transfers. Extensive data buffering in both directions increases system performance by maximizing PCI and EISA Bus efficiency and allowing concurrency on the two buses. The PCEB's buffer management mechanism ensures data coherency. The PCEB integrates central bus control functions including a programmable bus arbiter for the PCI Bus and EISA data swap logic for the EISA Bus. Integrated system functions include PCI parity generation, system error reporting, and programmable PCI and EISA memory and I/O address space mapping and decoding. The PCEB also contains a BIOS Timer that can be used to implement timing loops. The PCEB is intended to be used with the EISA System Component (ESC) to provide an EISA I/O subsystem interface.

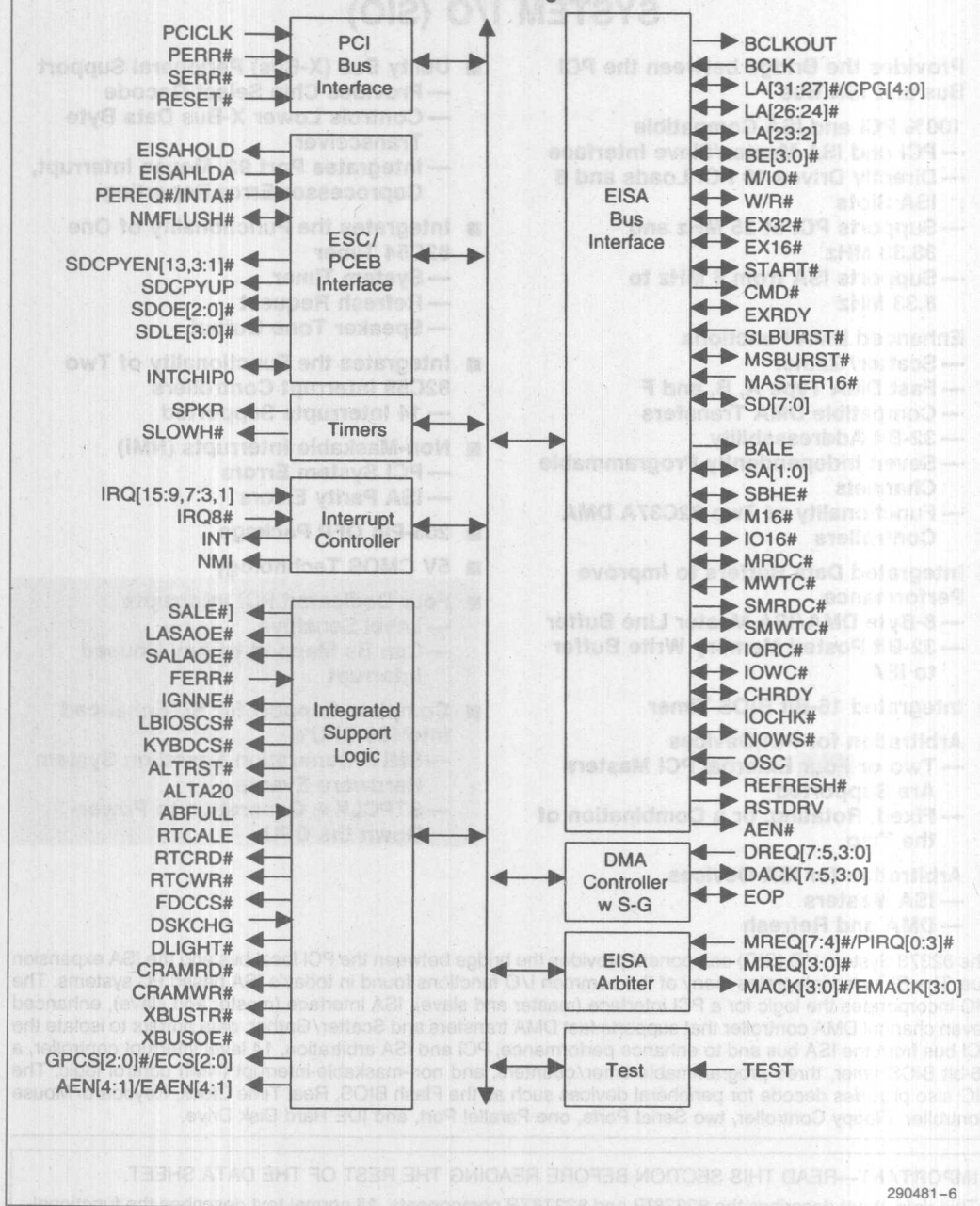


82374EB EISA SYSTEM CONTROLLER (ESC)

- **Integrates EISA Compatible Bus Controller**
 - Translates Cycles between EISA and ISA Bus
 - Supports EISA Burst and Standard Cycles
 - Supports ISA No Wait State Cycles
 - Supports Byte Assembly/Disassembly for 8-Bit, 16-Bit and 32-Bit Transfers
 - Supports Bus Frequency up to 8.33 MHz
- **Supports Eight EISA Slots**
 - Directly Drives Address, Data and Control Signals for Eight Slots
 - Decodes Address for Eight Slot Specific AENs
- **Provides Enhanced DMA Controller**
 - Provides Scatter-Gather Function
 - Supports Type A, Type B, Type C (Burst), and Compatible DMA Transfers
 - Provides Seven Independently Programmable Channels
 - Integrates Two 82C37A Compatible DMA Controllers
- **Provides High Performance Arbitration**
 - Supports Eight EISA Masters and PCEB
 - Supports ISA Masters, DMA Channels, and Refresh
 - Provides Programmable Arbitration Scheme for Fixed, Rotating, or Combination Priority
- **Integrates Support Logic for X-Bus Peripherals and More**
 - Generates Chip Selects/Encoded Chip Selects for Floppy and Keyboard Controller, IDE, Parallel/Serial Ports, and General Purpose Peripherals
 - Provides Interface for Real Time Clock
 - Generates Control Signals for X-Bus Data Transceiver
 - Integrates Port 92, Mouse Interrupt, and Coprocessor Error Reporting
- **Integrates the Functionality of Two 82C59 Interrupt Controllers and Two 82C54 Timers**
 - Provides 14 Programmable Channels for Edge or Level Interrupts
 - Provides 4 PCI Interrupts Routable to Any of 11 Interrupt Channels
 - Supports Timer Function for Refresh Request, System Timer, Speaker Tone, Fail Safe Timer, and Periodic CPU Speed Control
- **Generates Non-Maskable Interrupts (NMI)**
 - PCI System Errors
 - PCI Parity Errors
 - EISA Bus Parity Errors
 - Fail Safe Timer
 - Bus Timeout
 - Via Software Control
- **Provides BIOS Interface**
 - Supports 512 KBytes of Flash or EPROM BIOS on the X-Bus
 - Allows BIOS on PCI
 - Supports Integrated VGA BIOS
- **208-Pin QFP Package**
- **5V CMOS Technology**

The 82374EB EISA System Component (ESC) provides all the EISA system compatible functions. The ESC, with the PCEB, provides all the functions to implement an EISA to PCI bridge and EISA I/O subsystem. The ESC integrates the common I/O functions found in today's EISA based PC systems. The ESC incorporates the logic for an EISA (master and slave) interface, EISA Bus Controller, enhanced seven channel DMA controller with Scatter-Gather support, EISA arbitration, 14 channel interrupt controller, five programmable timer/counters, and non-maskable interrupt (NMI) control logic. The ESC also integrates support logic to decode peripheral devices such as the Flash BIOS, Real Time Clock, Keyboard/Mouse Controller, Floppy Controller, two Serial Ports, one Parallel Port, and IDE Hard Disk Drive.

ESC Block Diagram



2

290481-6

82378 SYSTEM I/O (SIO)

- Provides the Bridge between the PCI Bus and ISA Bus
- 100% PCI and ISA Compatible
 - PCI and ISA Master/Slave Interface
 - Directly Drives 10 PCI Loads and 6 ISA Slots
 - Supports PCI at 25 MHz and 33.33 MHz
 - Supports ISA from 6 MHz to 8.33 MHz
- Enhanced DMA Functions
 - Scatter/Gather
 - Fast DMA Type A, B, and F
 - Compatible DMA Transfers
 - 32-Bit Addressability
 - Seven Independently Programmable Channels
 - Functionality of Two 82C37A DMA Controllers
- Integrated Data Buffers to Improve Performance
 - 8-Byte DMA/ISA Master Line Buffer
 - 32-Bit Posted Memory Write Buffer to ISA
- Integrated 16-Bit BIOS Timer
- Arbitration for PCI Devices
 - Two or Four External PCI Masters Are Supported
 - Fixed, Rotating, or a Combination of the Two
- Arbitration for ISA Devices
 - ISA Masters
 - DMA and Refresh
- Utility Bus (X-Bus) Peripheral Support
 - Provides Chip Select Decode
 - Controls Lower X-Bus Data Byte Transceiver
 - Integrates Port 92, Mouse Interrupt, Coprocessor Error Reporting
- Integrates the Functionality of One 82C54 Timer
 - System Timer
 - Refresh Request
 - Speaker Tone Output
- Integrates the Functionality of Two 82C59 Interrupt Controllers
 - 14 Interrupts Supported
- Non-Maskable Interrupts (NMI)
 - PCI System Errors
 - ISA Parity Errors
- 208-Pin QFP Package
- 5V CMOS Technology
- Four Dedicated PCI Interrupts
 - Level Sensitive
 - Can Be Mapped to Any Unused Interrupt
- Complete Support for SL Enhanced Intel486 CPU's
 - SMI# Generation Based on System Hardware Events
 - STPCLK# Generation to Power-Down the CPU

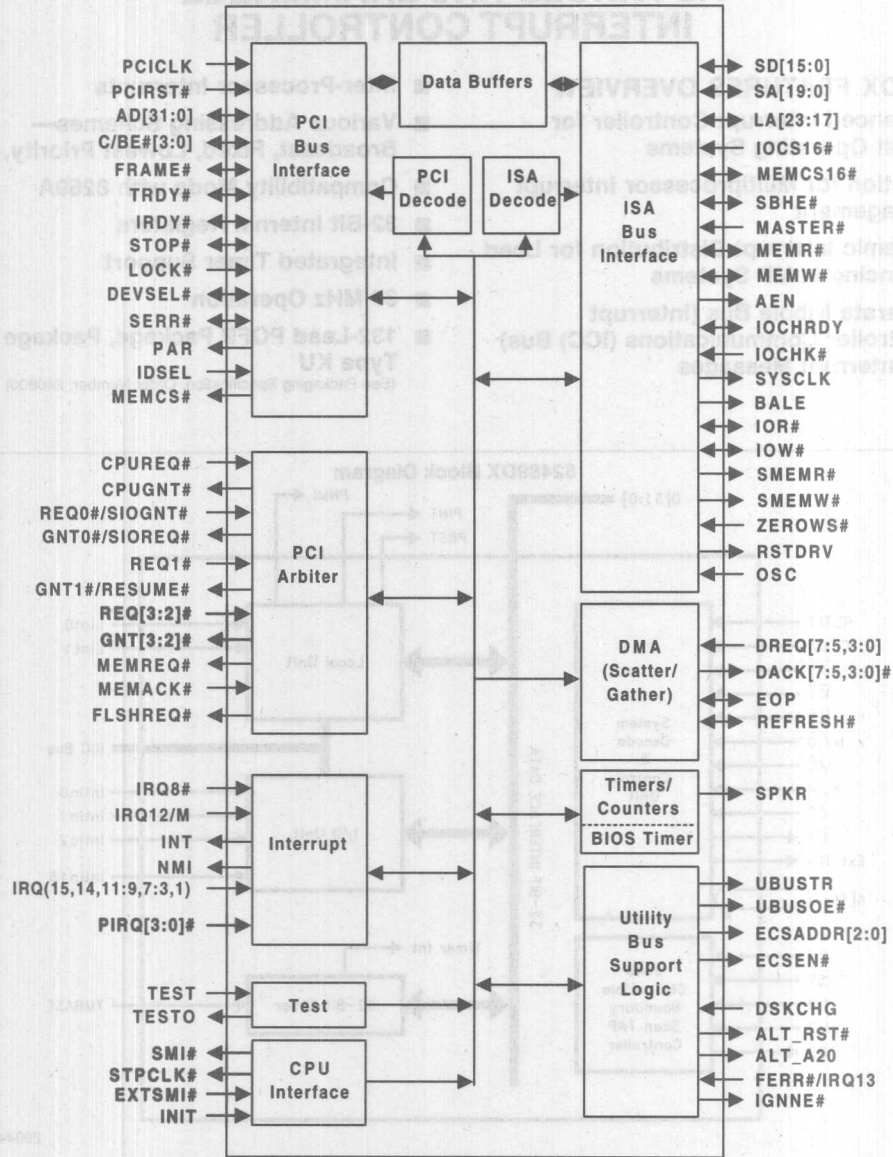
The 82378 System I/O (SIO) component provides the bridge between the PCI local bus and the ISA expansion bus. The SIO also integrates many of the common I/O functions found in today's ISA based PC systems. The SIO incorporates the logic for a PCI interface (master and slave), ISA interface (master and slave), enhanced seven channel DMA controller that supports fast DMA transfers and Scatter/Gather, data buffers to isolate the PCI bus from the ISA bus and to enhance performance, PCI and ISA arbitration, 14 level interrupt controller, a 16-bit BIOS timer, three programmable timer/counters, and non-maskable-interrupt (NMI) control logic. The SIO also provides decode for peripheral devices such as the Flash BIOS, Real Time Clock, Keyboard/Mouse Controller, Floppy Controller, two Serial Ports, one Parallel Port, and IDE Hard Disk Drive.

IMPORTANT—READ THIS SECTION BEFORE READING THE REST OF THE DATA SHEET.

This data sheet describes the 82378IB and 82378ZB components. All normal text describes the functionality for both components. All features that exist on the 82378ZB are shaded as shown below.

This is an example of what the shaded sections that apply only to the 82378ZB component look like.

SIO Block Diagram



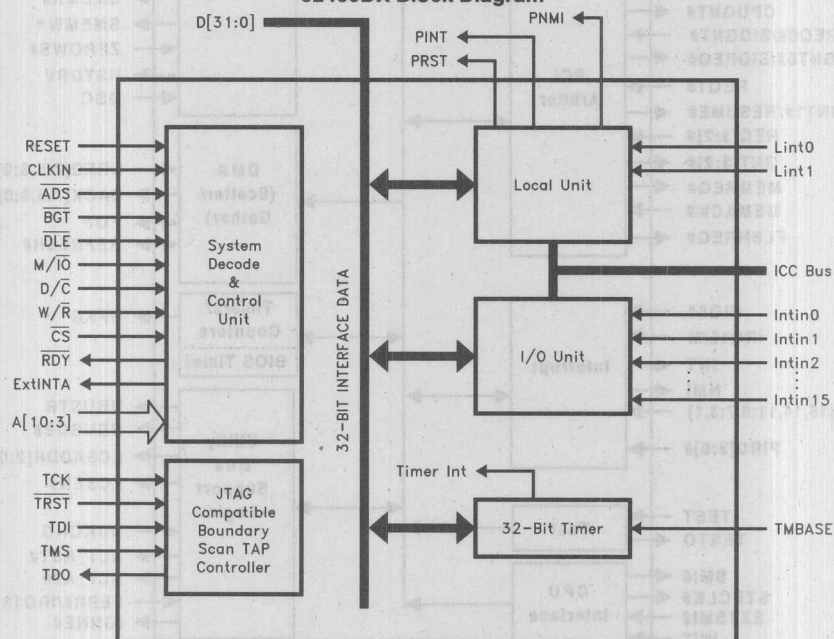
290481-7

82489DX ADVANCED PROGRAMMABLE INTERRUPT CONTROLLER

82489DX FEATURES OVERVIEW

- Advanced Interrupt Controller for 32-Bit Operating Systems
- Solution for Multiprocessor Interrupt Management
- Dynamic Interrupt Distribution for Load Balancing in MP Systems
- Separate Nibble Bus (Interrupt Controller Communications (ICC) Bus) for Interrupt Messages
- Inter-Processor Interrupts
- Various Addressing Schemes—Broadcast, Fixed, Lowest Priority, etc.
- Compatibility Mode with 8259A
- 32-Bit Internal Registers
- Integrated Timer Support
- 33 MHz Operation
- 132-Lead PQFP Package, Package Type KU
(See Packaging Specification, Order Number: 240800)

82489DX Block Diagram



290446-1

Refer to Application Note AP-388: 82489DX User's Manual (Order Number 292116) when evaluating your design needs.

82489DX

Advanced Programmable Interrupt Controller

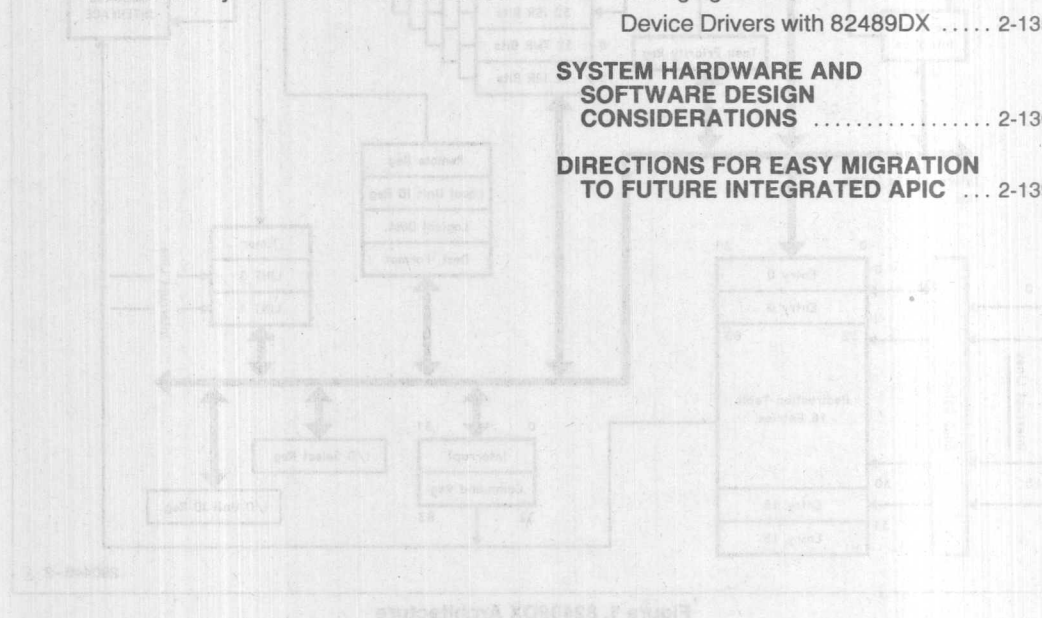
CONTENTS	PAGE
1.0 INTRODUCTION	2-62
2.0 FUNCTIONAL OVERVIEW	2-63
ICC Bus	2-63
Local Unit	2-63
I/O Unit	2-63
Timer	2-63
3.0 PIN DESCRIPTION	2-63
4.0 FUNCTIONAL DESCRIPTION	2-67
I/O Unit	2-67
Local Unit	2-68
5.0 INTERRUPT CONTROL MECHANISM	2-70
5.1 Interrupts	2-70
Total Allowed Interrupt Vectors	2-70
Interrupt Sources	2-71
Interrupt Destinations	2-71
Interrupt Delivery	2-71
5.2 Interrupt Redirection	2-72
Inter-82489DX Communication	2-72
6.0 82489DX LOCAL UNIT REGISTERS DESCRIPTION	2-72
6.1 Local Unit ID Register	2-72
82489DX Local Unit ID Register	2-72
6.2 Destination Format Register	2-72
6.3 Local Interrupt Vector Table Registers	2-73
Local Interrupts 0,1 Interrupt Vectors	2-73
6.4 Inter-Processor Interrupt Registers	2-74
Interrupt Command Register [31:0]	2-74

CONTENTS	PAGE
Interrupt Command Register [63:32]	2-76
6.5 IRR, ISR, TMR Registers	2-76
Interrupt Acceptance	2-76
Acceptance Mechanism	2-78
6.6 Tracking Processor Priority	2-80
Task Priority Register	2-80
6.7 Dispensing Interrupts	2-81
Dispensing Interrupts to the Local Processor	2-81
6.8 Spurious Interrupt Vector Register	2-81
Spurious Interrupt	2-81
Unit Enable	2-81
6.9 End-Of-Interrupt (EOI) Register	2-81
6.10 Remote Read Register	2-82
6.11 82489DX Local Configuration Local Version Register	2-82
6.12 82489DX Timer Registers	2-82
Overview	2-82
Time Base	2-82
Timer	2-83
Timer Vector Table	2-83
7.0 82489DX I/O UNIT REGISTERS	2-84
Registers Addressing Scheme	2-84
82489DX I/O Unit Configuration	2-85
I/O Unit ID Register	2-85
I/O Unit Version Register	2-85
I/O Unit Interrupt Source Registers	2-85
Redirection Tables	2-85
Descriptions	2-86
Destination	2-87

CONTENTS	PAGE	CONTENTS	PAGE
8.0 ICC BUS DEFINITION	2-88	Pause-IR	2-111
Physical Characteristics	2-88	Exit 2-IR	2-111
Bus Arbitration	2-88	Update-IR	2-111
Lowest-Priority Arbitration	2-89	Instruction Register	2-112
ICC Bus Message Formats	2-89	Bypass Instruction	2-112
Long Message Format	2-91	Exttest Instruction	2-112
9.0 HARDWARE TIMINGS	2-92	Sample/Preload Instruction	2-112
Interfacing to the ICC Bus	2-93	dcode Instruction	2-112
First Order Buffer Models	2-93	Device Identification Register (DID) ..	2-112
MBO Pull-Up Register	2-93	Boundary Scan Register	2-112
Driving Lumped Capacitance	2-93	Boundary Scan Cell Names in Order from tdi to tdo	2-114
Driving Transmission Lines	2-94	Bypass Register	2-116
External Drivers/Buffered ICC Bus	2-96	JTAG TAP Controller Initialization	2-116
Transmission Line Termination	2-98	11.0 ELECTRICAL CHARACTERISTICS	2-117
ICC Bus Operating Frequency	2-98	11.1 D.C. Specifications	2-117
9.1 82489DX Register Access		11.2 A.C. Specifications	2-118
Timing	2-100	12.0 REGISTER SUMMARY	2-120
Timing Diagram Notation	2-100	I/O Unit Registers	2-121
Register WRITE Timing	2-101	Local Unit Registers	2-122
Register READ Timing	2-106	13.0 TIMING DIAGRAMS	2-123
Interrupt Acknowledge Timing	2-106	14.0 PACKAGE PIN-OUT	2-127
Reset and Miscellaneous Timing	2-107	15.0 PACKAGE THERMAL SPECIFICATION	2-128
10.0 BOUNDARY SCAN DESCRIPTION	2-107	16.0 GUIDELINES FOR 82489DX USERS	2-129
10.1 Boundary Scan Architecture	2-107	16.1 Initialization	2-129
Test Access Ports	2-108	16.2 Compatibility	2-129
TAP Controller	2-108	Compatibility Levels	2-129
Test-Logic-Reset	2-109	82489DX/8259A Interaction	2-129
Run-Test/Idle	2-110	82489DX/8259A Dual Mode Connection	2-130
Select-DR-Scan	2-110	16.3 Hardware Guidelines	2-131
Select-IR-Scan	2-110	82489DX Hardware State on Reset	2-131
Capture-DR	2-110	Pull Up and Pull Down Resistors ..	2-131
Shift-DR	2-110	Pint and ExINTA Timings	2-131
Exit 1-DR	2-110	ExtINTA Timings	2-131
Pause-DR	2-110		
Exit 2-DR	2-110		
Update-DR	2-111		
Capture-IR	2-111		
Shift-IR	2-111		
Exit 1-IR	2-111		

CONTENTS	PAGE
82489DX and Memory Mapping ..	2-131
JTAG Circuit Considerations	2-131
16.4 Programming Guidelines	2-132
Unique ID Requirement	2-132
Atomic Write Read to Task Priority Register	2-132
Critical Regions and Mutual Exclusion	2-132
Interrupt Command Register Programming Sequence	2-132
Interrupt Vector	2-132
Local and I/O Unit	2-132
ICR (Interrupt Command Register)	2-132
ISR/IRR/TMR	2-133
Focus Processor	2-133
ExtINT Interrupt Posting	2-133
Synchronizing Arb IDs	2-133
Lowest Priority	2-133

CONTENTS	PAGE
Disabling Local Unit	2-133
Issuing EOI	2-134
External Interrupts and EOI	2-134
Spurious Interrupts and EOI	2-134
NMI and EOI	2-134
Task Priority Register	2-134
ExtINT Interrupt and Task Priority	2-134
Removing Masks	2-134
Delivery Mode and Trigger Mode	2-134
Assigning Interrupt Vectors	2-135
Sending Inter-Processor Interrupts	2-135
Delay with Level Triggered Interrupts	2-135
Reset Deassert	2-135
Interrupt Masking	2-135
Changing Redirection Tables	2-135
Device Drivers with 82489DX	2-135
SYSTEM HARDWARE AND SOFTWARE DESIGN CONSIDERATIONS	2-136
DIRECTIONS FOR EASY MIGRATION TO FUTURE INTEGRATED APIC	2-139



1.0 INTRODUCTION

The 82489DX Advanced Programmable Interrupt Controller provides multiprocessor interrupt management, providing both static and dynamic symmetrical interrupt distribution across all processors.

The main function of the 82489DX is to provide interrupt management across all processors. This dynamic interrupt distribution includes routing of the interrupt to the lowest-priority processor. The 82489DX works in systems with multiple I/O subsystems, where each subsystem can have its own set of interrupts. This chip also provides inter-processor interrupts, allowing any processor to interrupt any processor or set of processors. Each 82489DX I/O unit Interrupt Input pin is individually programmable by software as either edge or level triggered. The interrupt vector and interrupt steering information

can be specified per pin. A 32-bit wide timer is provided that can be programmed to interrupt the local processor. The timer can be used as a counter to provide a time base to software running on the processor, or to generate time slice interrupts locally to that processor. The 82489DX provides 32-bit software access to its internal registers. Since no 82489DX register reads have any side effects, the 82489DX registers can be aliased to a user read-only page for fast user access (e.g., performance monitoring timers).

The 82489DX supports a generalized naming/addressing scheme that can be tailored by software to fit a variety of system architectures and usage models. It also supports 8259A compatibility by becoming virtually transparent with regard to an externally connected 8259A style controller, making the 8259A visible to software.

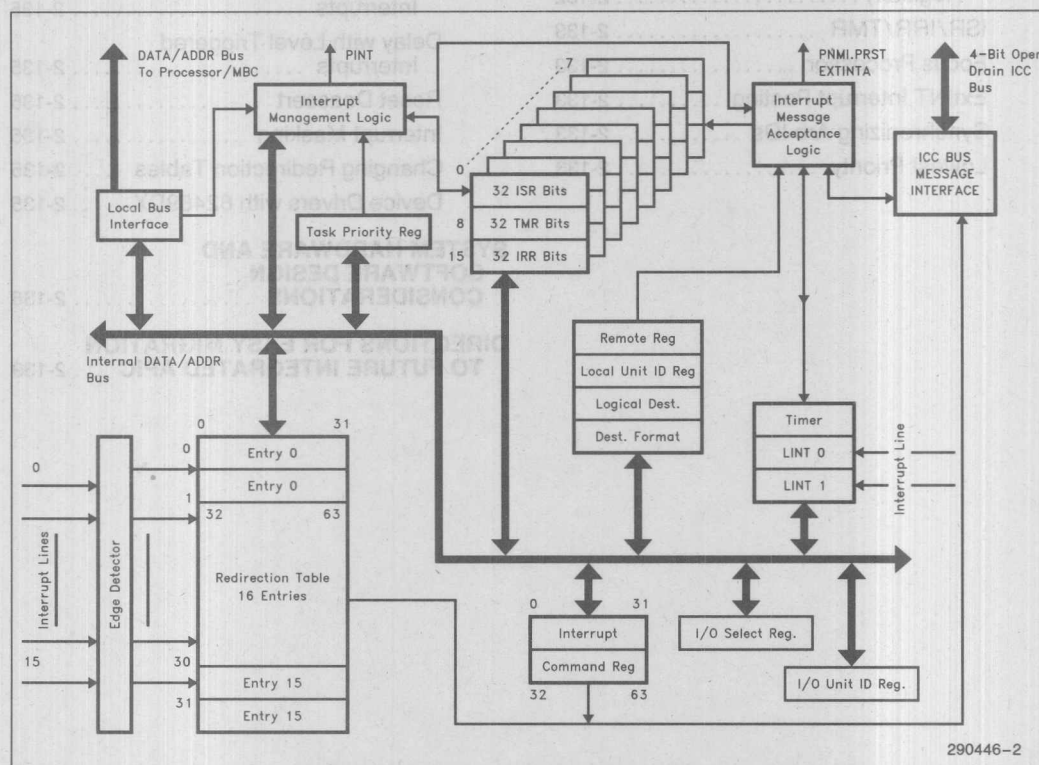


Figure 1. 82489DX Architecture

2.0 FUNCTIONAL OVERVIEW

82489DX Functional Blocks

82489DX contains one Local Unit, one I/O unit and a timer. The ICC bus is used to pass interrupt messages.

ICC BUS

The ICC bus is a 5-wire synchronous bus connecting all 82489DXs (all I/O Units and all Local Units). The Local Units and I/O Units communicate over this ICC bus. Four of these five wires are used for data transmissions and arbitration, and one wire is a clock.

LOCAL UNIT

The Local Unit contains the necessary intelligence to determine whether or not its processor should accept interrupt messages sent on the ICC bus by other Local Units and I/O Units. The Local Unit also provides local pending of interrupts, nesting and masking of interrupts, and handles all interactions with its local processor such as the INT/INTA/EOI protocol. The Local Unit further provides inter-processor interrupt functionality and a timer to its local processor. The interface of a processor to its 82489DX Local Unit is identical for every processor.

I/O UNIT

The I/O Unit provides the interrupt input pins on which I/O devices inject interrupts into the system in

the form of an edge or a level. The I/O unit also contains a Redirection Table for the interrupt input pins. Each entry in the Redirection Table can be individually programmed to indicate whether an interrupt on the pin is recognized as either an edge or a level; what vector and also what priority the interrupt has; and which of all possible processors should service the interrupt and how to select that processor (statically or dynamically). The information in the table is used to send interrupt messages to all 82489DX Units via the ICC bus.

TIMER

The 82489DX provides a 32-bit wide timer that can be programmed to interrupt the local processor. The timer can be used as a counter to provide a time-base to software running on the processor, or to generate time-slice interrupts local to that processor.

3.0 PIN DESCRIPTION

The 82489DX pin description is organized in a small number of functional groups. Pin definitions and protocols have been designed to minimize interface issues. In particular, they support the notion of independently controlled address and data phases. The primary host interface is synchronous in nature.

In the following pin definition table if the signal name has () over it, the signal is in its active state when it has a low level. The signal direction column identifies output only signals as a continuous drive (O), tristate (T/S), or open drain (O/D). All bi-directional (BI-D) signals have tri-stating outputs.

Pin	Signal	Direction	Notes
80	INT0	O	Two LOCAL INTERRUPT INPUT pins accept edge or level sensitive interrupt requests that can only be delivered to the connected processor. These pins are active high.
81	INT1	O	
82	INT2	O	
83	INT3	O	
84	INT4	O	
85	INT5	O	
86	INT6	O	
87	INT7	O	
88	INT8	O	
89	INT9	O	
90	INT10	O	
91	INT11	O	
92	INT12	O	
93	INT13	O	
94	INT14	O	
95	INT15	O	
96	INT16	O	
97	INT17	O	
98	INT18	O	
99	INT19	O	
100	INT20	O	
101	INT21	O	
102	INT22	O	
103	INT23	O	
104	INT24	O	
105	INT25	O	
106	INT26	O	
107	INT27	O	
108	INT28	O	
109	INT29	O	
110	INT30	O	
111	INT31	O	

Pin Definition Table

Symbol	Pin No.	Type	Function
SYSTEM PINS			
RESET	65	I	The RESET INPUT forces 82489DX to enter its initial state. The 82489DX Local Unit in turn asserts its PRST (Processor Reset) output. All tri-state outputs remain in high impedance until explicitly enabled.
ExtINTA	41	O	The EXTERNAL INTERRUPT ACKNOWLEDGE output is asserted (high) when an external interrupt controller (e.g., 8259) is expected to respond to the current INTA cycle. If deasserted (low), 82489DX will respond, and the INTA cycle must not be delivered to the external controller.
CLKIN	57	I	CLOCK INPUT provides reference timing for most of the bus signals.
TRST	56	I	TEST RESET is the JTAG compatible boundary scan TAP controller reset pin. A weak pull-up keeps the pin high if not driven.
TCK	55	I	TEST CLOCK is the clock input for the JTAG compatible boundary scan controller and latches.
TDI	53	I	TEST DATA INPUT is the test data input pin for the JTAG compatible boundary scan chain and TAP controller. A weak pull-up keeps this pin high if not driven.
TDO	52	O	TEST DATA OUTPUT is the test data output for the JTAG compatible boundary scan chain.
TMS	54	I	TEST MODE SELECT is the test mode select pin for the JTAG boundary scan TAP controller. A weak pull-up keeps this pin high if not driven.
TIMER PIN			
TMBASE	59	I	The TIME BASE input provides a standard frequency that is only used by the 82489DX timer and that is independent of the system clock.
INTERRUPT PINS			
INTIN[15:0]	82-97	I	These 16 INTERRUPT INPUT pins accept edge or level sensitive interrupt requests from I/O or other devices. The pin numbers are specified respectively. INTIN15 corresponds to pin number 82, INTIN14 corresponds to pin number 83 etc., and INTIN0 corresponds to pin number 97. These pins are active high.
LINTIN[1] LINTIN[0]	80 81	I I	Two LOCAL INTERRUPT INPUT pins accept edge or level sensitive interrupt requests that can only be delivered to the connected processor. These pins are active high.
REGISTER ACCESS PINS			
ADS	64	I	ADDRESS STROBE signal indicating the start of a bus cycle. 82489DX does not commit to start the cycle internally until BUS GRANT is detected active.

Pin Definition Table (Continued)

Symbol	Pin No.	Type	Function
REGISTER ACCESS PINS (Continued)			
M/ $\overline{\text{IO}}$, D/ $\overline{\text{C}}$, W/ $\overline{\text{R}}$	63 61 62	I I I	Bus cycle definition signals. Note that since the 82489DX registers can be mapped in either memory or I/O space, the M/ $\overline{\text{IO}}$ pin is not used for register access cycles; it is only used to decode interrupt acknowledge cycles. 82489DX does not respond to code read cycles.
BGT	66	I	The BUS GRANT input is optional and is used to indicate the address phase of a bus cycle in configurations where address timing cannot be inferred from $\overline{\text{ADS}}$. This signal is really used as an address latch enable, but is named as it is to indicate that it can normally be connected to the Intel Cache Controller generated signal of the same name. Must be tied low if not used.
$\overline{\text{CS}}$	74	I	The CHIP SELECT input indicates that the 82489DX registers are being addressed.
A3 A4 A5 A6 A7 A8 A9 A10	31 29 28 27 26 24 22 21	BI-D BI-D BI-D BI-D BI-D BI-D BI-D BI-D	The address pins are used as inputs in addressing internal register space. Output function is reserved. They are also used to latch local unit ID on reset.
$\overline{\text{DLE}}$	73	I	DATA LATCH/ENABLE is optional and is used to indicate committing the data phase of a bus cycle in configurations where data timing cannot be inferred from other cycle timings. Must be tied low if not used.
D31 D30 D29 D28 D27 D26 D25 D24 D23 D22 D21 D20 D19 D18 D17 D16 D15 D14 D13 D12 D11	105 107 109 110 111 112 114 115 116 118 119 121 122 123 124 125 128 129 130 131 2	BI-D BI-D	The DATA BUS is for all register accesses and interrupt vectoring.

2

Pin Definition Table (Continued)

Symbol	Pin No.	Type	Function
RESERVED PINS			
Reserved	34, 42	NC	These pins MUST BE LEFT OPEN .
Reserved	70, 72, 75		Reserved by Intel. These pins should be strapped to V_{CC}.
Reserved	71, 19, 20		Reserved by Intel. These pins should be strapped to GND.
POWER AND GROUND PINS			
V _{CC}	1, 32, 69, 98	POWER	Nominally +5V. These pins along with V _{SS} and V _{SSI} should be separately bypassed.
V _{CCP}	6, 15, 25, 100, 108, 117, 126	POWER	Nominally +5V. These pins along with V _{SSP} should be separately bypassed.
V _{CCPO}	39, 46	POWER	Nominally +5V. These pins along with V _{SSPO} should be separately bypassed.
V _{SS}	5, 33, 67, 68, 99	GND	Nominally 0V. These pins along with V _{CC} should be separately bypassed.
V _{SSP}	10, 17, 23, 30, 106, 113, 120, 127, 132,	GND	Nominally 0V. These pins along with V _{CCP} should be separately bypassed.
V _{SSPO}	36, 40, 44, 47, 50	GND	Nominally 0V. These pins along with V _{CCPO} should be separately bypassed.
V _{SSI}	58	GND	Nominally 0V. These pins along with V _{CC} should be separately bypassed.

NOTE:

V_{CC}, V_{CCP} and V_{CCPO} should be of same voltage. V_{SS}, V_{SSP}, V_{SSPO} and V_{SSI} should be 0V.

4.0 FUNCTIONAL DESCRIPTION

As far as interrupt management is concerned, the 82489DX's interrupt control function spans over two functional units, the I/O Unit of which there is one per I/O subsystem, and the Local Unit of which there is one per processor. 82489DX has one I/O unit and one Local Unit in a single package. This section takes a detailed look at both local and I/O Units.

I/O Unit

The I/O Unit consists of a set of Interrupt Input pins, an Interrupt Redirection Table, and a message unit for sending and receiving messages from the ICC bus. The I/O Unit is where I/O devices inject their interrupts, the I/O Unit selects the corresponding entry in the Redirection Table and uses the information in that entry to format an interrupt request message. The message unit then broadcasts this message over the ICC bus. The content of the Redirection Table is under software control and is assigned benign defaults upon reset. The masks in the Redirection Table entries are set to 1 at *hardware reset* to disable the interrupts.

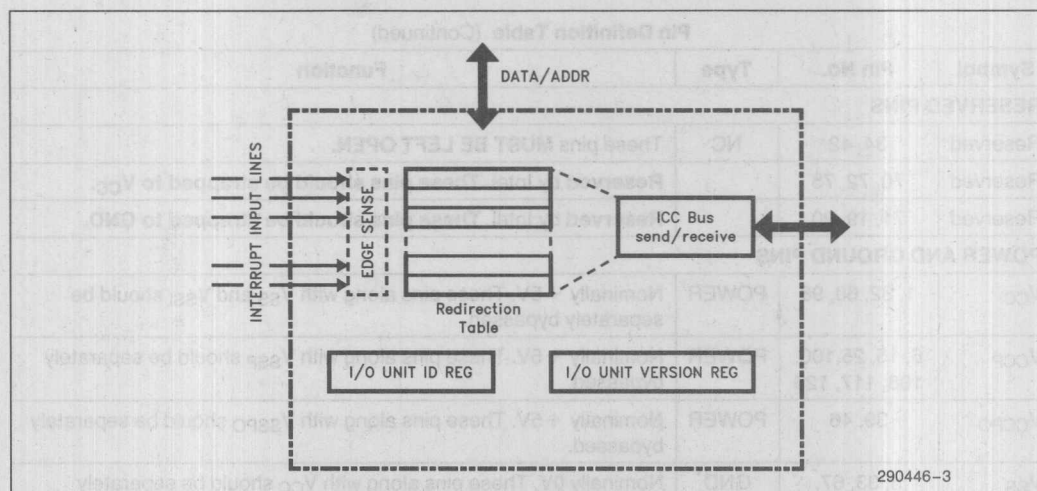


Figure 2. 82489DX I/O Unit Block Diagram

Local Unit

Interrupt Management of the Local Unit is responsible for local interrupt sources, interrupt acceptance, dispensing interrupts to the processor, and sending inter-processor interrupts. Depending on the delivery

mode of the interrupt, zero, one or more units can accept an interrupt. A Local Unit accepts an interrupt only if it will deliver the interrupt to its processor. Accepting an interrupt is purely an inter-82489DX matter; dispensing an interrupt to the local processor only involves a 82489DX and its local processor.

The I/O Unit consists of a set of interrupt input pins, an interrupt Redirection Table, and a message unit for sending and receiving messages from the ICC bus. The I/O Unit is where I/O devices inject their interrupts. The I/O Unit selects the corresponding entry in the Redirection Table and uses the information in that entry to format an interrupt request message. The message unit then processes the message over the ICC bus. The format of the Redirection Table is under software control and is as signed random data on reset. The entries in the Redirection Table are set to 1 at hardware reset to enable the interrupts.

As far as interrupt management is concerned, the 82489DX's interrupt control function space over two functional units: the I/O Unit of which there is one per I/O processor, and the Local Unit of which there is one per processor. 82489DX has one I/O unit and one Local Unit in a single package. This section takes a detailed look at both local and I/O units.

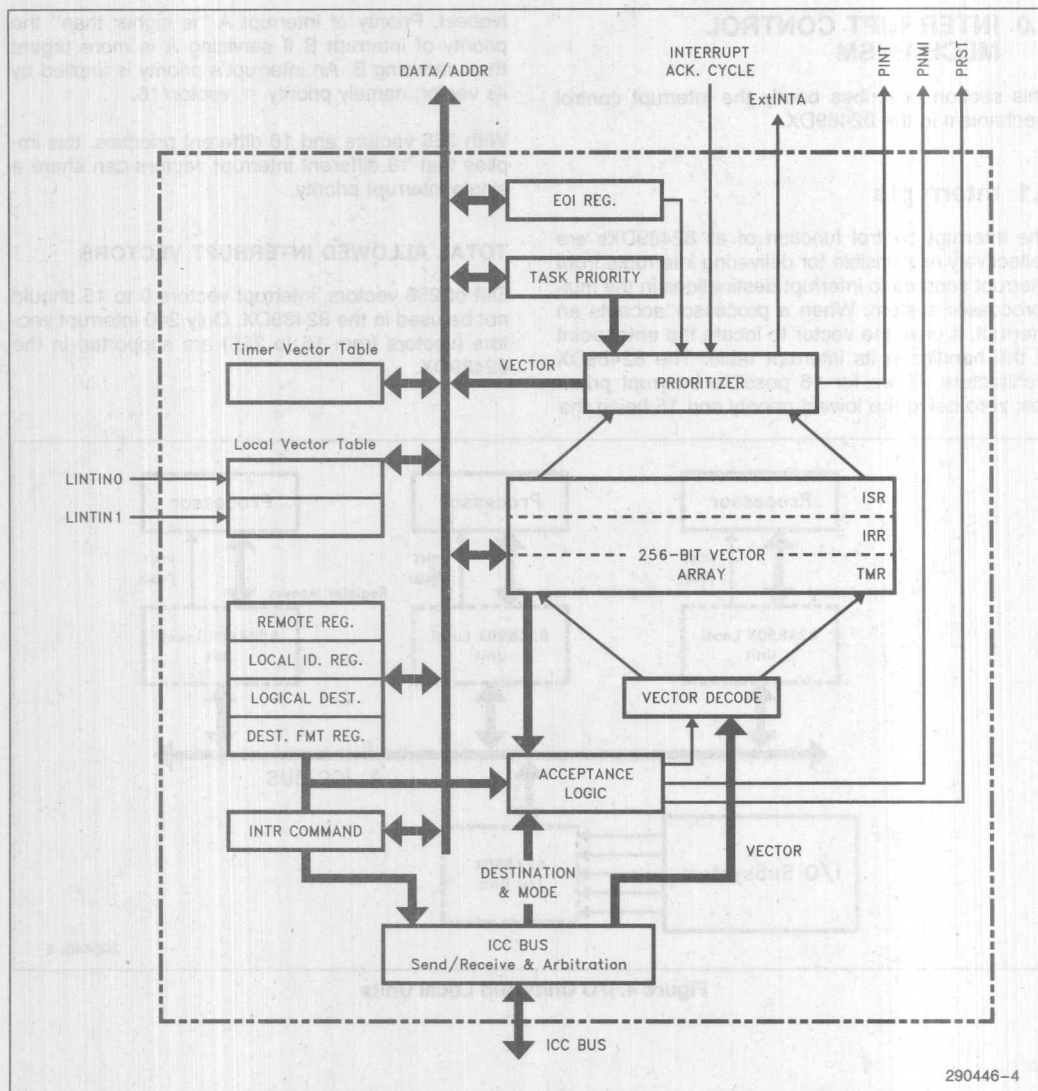


Figure 3. 82489DX Local Unit Block Diagram

5.0 INTERRUPT CONTROL MECHANISM

This section describes briefly the interrupt control mechanism in the 82489DX.

5.1 Interrupts

The interrupt control function of all 82489DXs are collectively responsible for delivering interrupts from interrupt sources to interrupt destinations in the multiprocessor system. When a processor accepts an interrupt, it uses the vector to locate the entry point of the handler in its interrupt table. The 82489DX architecture allows for 16 possible interrupt priorities; zero being the lowest priority and 15 being the

highest. Priority of interrupt A "is higher than" the priority of interrupt B if servicing A is more urgent than servicing B. An interrupt's priority is implied by its vector; namely $\text{priority} = \text{vector}/16$.

With 256 vectors and 16 different priorities, this implies that 16 different interrupt vectors can share a single interrupt priority.

TOTAL ALLOWED INTERRUPT VECTORS

Out of 256 vectors, interrupt vectors 0 to 15 should not be used in the 82489DX. Only 240 interrupt vectors (vectors from 16 to 255) are supported in the 82489DX.

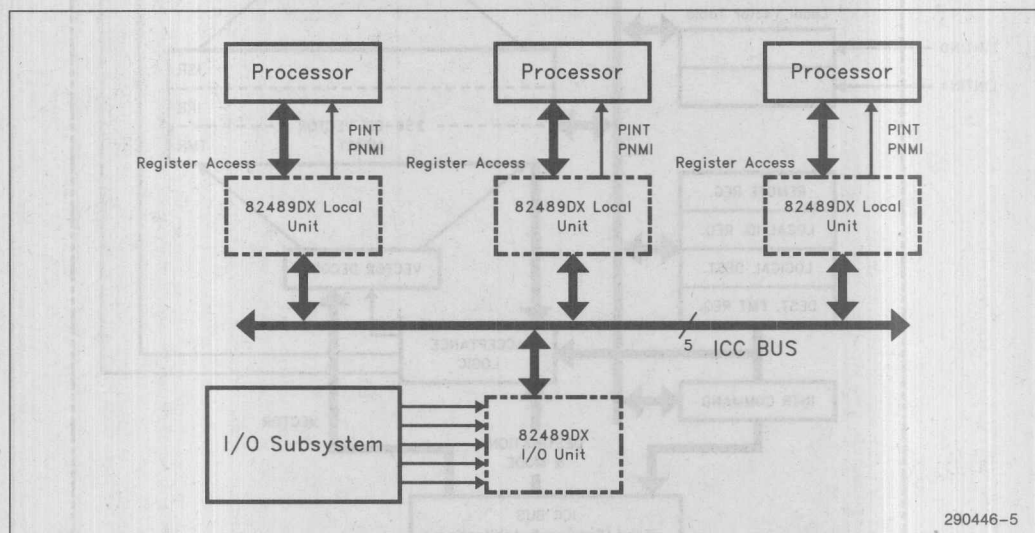


Figure 4. I/O Units and Local Units

INTERRUPT SOURCES

Interrupts are generated by a number of different interrupt sources in the system.

Possible interrupt sources are:

- Externally connected (I/O) devices. Interrupts from these external sources manifest themselves as edges or levels on interrupt input pins and can be redirected to any processor.
- Locally connected devices. These originate as edges or levels on interrupt pins, but they are always directed to the local processor only.
- 82489DX timer generated interrupts. Like locally connected devices, 82489DX timer can only interrupt its local processor.
- Processors. A processor can interrupt any individual processor or sets of processors. This supports software self-interrupts, preemptive scheduling, TLB flushing, and interrupt forwarding. A processor generates interrupts by writing to the interrupt command register in its Local Unit.

INTERRUPT DESTINATIONS

I/O Units can only source interrupts whereas Local Units can both source and accept interrupts, so whenever "interrupt destination" is discussed, it is implied that the Local Unit is the destination of the interrupt. In physical mode the destination processor is specified by a unique 8-bit 82489DX local ID. Only a single destination or a broadcast to all (LOCAL ID of all ones) can be specified in physical destination mode.

In logical mode destinations are specified using a 32-bit destination field. All Local Units contain a 32-bit Logical Destination register against which the destination field of the interrupt is matched to determine if the receiver is being targeted by the interrupt. An additional 32-bit Destination Format register in each Local Unit enables the logical mode addressing.

INTERRUPT DELIVERY

The description of interrupt delivery makes frequent use of the following terms:

- Each processor has a processor priority that reflects the relative importance of the code the processor is currently executing. This code can be part of a process or thread, or can be an interrupt handler. A processor's priority fluctuates as a processor switches threads, a thread or handler raises and lowers its priority level to mask out interrupt, and the processor enters an interrupt handler and returns from an interrupt handler to previously interrupted activity.

- A processor is lowest priority within a given group of processors if its processor priority is the lowest of all processors in the group. Note that more than one processor can be the lowest priority in a given group.
- A processor is the focus of an interrupt if it is currently servicing that interrupt, or if it currently has a request pending for the interrupt.

Interrupt delivery begins with an interrupt source injecting its interrupt into the interrupt system at one of the 82489DX. Delivery is complete only when the servicing processor tells its 82489DX Local Unit it is complete by issuing an end-of-interrupt (EOI) command to its 82489DX Local Unit. Only then has all (relevant) internal state regarding that occurrence of the interrupt been erased. The interrupt system guarantees exactly-once delivery semantics of interrupts to the specified destinations. Exactly-once guaranteed delivery implies a number of things:

- The interrupt system never rejects interrupts; it never NAKs interrupt injection, interrupts are never lost, and the same interrupt (occurrence) is never delivered more than once.

Clearly a single edge interrupt or level interrupt counts as a single occurrence of an interrupt. In uniprocessor systems, an occurrence of an interrupt that is already pending (IRR) cannot be distinguished from the previous occurrence. All occurrences are recorded in the same IRR bit. They are therefore treated as "the same" interrupt occurrence.

For lowest-priority delivery mode, by delivering an interrupt first to its focus processor (if it currently has one), the identical behavior can be achieved in a MP (Multiprocessor) system. If an interrupt has a focus processor then the interrupt will be delivered to the interrupt's focus processor independent of priority information. This means that even if there is a lower priority processor compared to the focus processor, the interrupt still gets delivered to the focus processor.

Each edge occurring on an edge triggered interrupt input pin is clearly a one-shot event; each occurrence of an edge is delivered. An active level on a level triggered interrupt input pin represents more of a "continuous event". Repeatedly broadcasting an interrupt message while the level is active would cause flooding of the ICC bus, and in effect transmits very little useful information since the same processor (the focus) would have to be the target.

Instead, for level triggered interrupts the 82489DX merely recreate the state of the interrupt input pin at the destination. The source 82489DX accomplishes this by tracking the state of the appropriate destination.

tion 82489DX's Interrupt Request Register (or pending bit) and only sending inter-82489DX messages when the state of the interrupt input pin and the destination's interrupt request enter a disagreement. Unlike edge triggered interrupts, when a level interrupt goes into service, the interrupt request at the servicing 82489DX is not automatically removed. If the handler of a level sensitive interrupt executes an EOI then that interrupt will immediately be raised to the processor again, unless the processor has explicitly raised its task priority, or the source of that interrupt has been removed.

5.2 Interrupt Redirection

This section specifically talks about how a processor is picked during interrupt delivery. The 82489DX supports two modes for selecting the destination processor: Fixed and Lowest Priority.

- **Fixed Delivery Mode**

In fixed delivery mode, the interrupt is unconditionally delivered to all local 82489DXs that match in the destination information supplied with the interrupt. Note that for I/O device interrupts typically only a single 82489DX would be listed in the destination. Priority and focus information are ignored. If the priority of a destination processor equal to or higher than the priority of the interrupt, then the interrupt is held pending locally in the destination processor's Local Unit, until the processor priority becomes low enough at which time the interrupt is dispensed to the processor. More than one processor can be the destination in fixed-delivery mode.

- **Lowest Priority Delivery Mode**

Under the lowest priority delivery mode, the processor to handle the interrupt is the one in the specified destination with the lowest processor priority value. If more than one processor is at the lowest priority, then a unique arbitration ID is used to break ties. For lowest priority dynamic delivery, the interrupt will always be taken by its focus processor if it has one. The lowest priority delivery method assures minimum interruption of high priority tasks. Since each Local Unit only knows its own processor priority, determining the lowest priority processor is done by arbitration on the ICC bus. Only one processor can be the destination in lowest-priority delivery mode.

INTER-82489DX COMMUNICATION

All I/O and Local Units communicate during interrupt delivery. Interrupt information is exchanged between different units on a dedicated five wire ICC bus in the form of broadcast messages. A 82489DX Unit's 8-bit ID is used as its name for the purpose of using the ICC bus, and all 82489DX units using one ICC bus should be assigned a different ID. The Arbitration

ID of the Local Units used to resolve ties during lowest priority arbitration is also derived from the Local Unit's ID.

6.0 82489DX LOCAL UNIT REGISTERS DESCRIPTION

6.1 Local Unit ID Register

Each 82489DX Local Unit has a register that contains the Local Unit's 8-bit ID. The Local Unit ID serves as a physical name of the 82489DX Local Unit. It can be used in specifying destination information and is also used for accessing the ICC bus. Eight address lines A[10]–A[3] are sampled on every clock edge while RESET is asserted. The last sample remains in the Local Unit ID register after reset. Alternatively, the ID can be loaded with a register write as part of software initialization. The Local Unit ID is read-write by software.

Bits [31..24]	Bits [23..0]
---------------	--------------

82489DX Local Unit ID Register

Bits [31..24] Local Unit ID: The Local Unit ID serves as the physical "name" of the Local Unit used for addressing the 82489DX in physical destination mode and for the ICC bus usage. In a system with say four 82489DX, there are 4 Local Units and 4 I/O Units. All the 8 units should be assigned different IDs. For future compatibility use only IDs from 0 to 14.

Bits [23..0] Bits [23..0] are Reserved. They should be written with 0.

6.2 Destination Format Register

Interrupt Destination can be either addressed physically or logically. When the interrupt message addresses the destination physically, each 82489DX in the ICC bus compares the address with its own unit ID. If the message is a broadcast type then every Local Unit accepts the interrupt.

When the message addresses the destination using logical addressing scheme each Local Unit in the ICC bus compares the logical address in the interrupt message with its own Logical Destination Register. If there is a bit match (i.e., if at least one of the corresponding pair of bits match) this local unit is selected for delivery.

All the 32 bits of Destination Format Register of all 82489DX connected in the ICC bus should be written with "1" to enable the addressing scheme.

Destination Format Register

Bits [31:0]

Logical Destination Register

Bits [31:0]

For future compatibility, use only bits 31–24 of Logical Destination Register. For binary compatibility, it is strongly recommended that 82489DX software use only 8 MSB of Logical Destination Register.

6.3 Local Interrupt Vector Table Registers

The Redirection Table serves to steer interrupts that originate in the I/O subsystems to the processors. The Local Vector Table is its equivalent for interrupts that are restricted to only the local processor. The Local Vector Table contains three 32-bit registers. Register 0 corresponds to the timer, registers 1 and 2 correspond to local interrupt input pins, LINTINO and LINTIN1.

The format of both the Local 0 and Local 1 interrupt vector tables are identical. The following register description talks about both Local Interrupts 0,1 vectors.

Local Interrupts 0, 1 Interrupt Vectors

Vector: [Bits 7–0]

This is the vector to use when generating an interrupt for this entry.

Delivery Mode: [Bits 10–8]

- 000: Fixed
- 001: <reserved>
- 010: <reserved>
- 011: <reserved>
- 100: NMI
- 101: <reserved>

Local INTO Vector Table								
Bits [31:17]	Bit 16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bits [10:8]	Bits [7:0]
Local INT1 Vector Table								
Bits [31:17]	Bit 16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bits [10:8]	Bits [7:0]

Figure 5. Local Vector Table

110: <reserved>

111: ExtINT

000: (fixed) means deliver the signal on the INT pin of the local processor. Trigger mode for "fixed" Delivery Mode can be edge or level.

100: (NMI) means deliver the signal on the NMI pin of the local processor. Vector information is ignored. A Delivery Mode equal to "NMI" requires a "level" triggered mode.

111: (ExtINTA) means deliver the signal to the INT pin of the local processor as an interrupt that originated in an externally connected (8259A compatible) interrupt controller. ExtINTA pin is asserted also. The INTA cycle that corresponds to this ExtINTA delivery, should be routed to the external controller that is expected to supply the vector. A delivery mode of ExtINT requires an edge trigger mode. (See the section on compatibility for more details.)

Bit 11: Bit 11 is Reserved. It should be written 0.

Delivery Status: [Bit 12]

This field is software-read only. Software writes to this field (as part of a 32-bit word) have no effect on this bit. Delivery status is a 1-bit field that contains the current status of the delivery of this interrupt. Two states are defined.

0: (idle) means that there is currently no activity for this interrupt.

1: (send pending) indicates that the interrupt has been injected, but its delivery is temporarily held up by the recently injected interrupts that are in the process of being delivered.

Bit 13: Bit 13 is Reserved. It should be written 0.

Remote IRR: [Bit 14]

This bit is used for level triggered local interrupts; its meaning is undefined for edge triggered interrupts. Remote IRR mirrors the interrupt's IRR bit of this local unit. Remote IRR is software read-only; software writes to this bit do not affect it.

Trigger Mode: [Bit 15]

The Trigger Mode field indicates the type of signal on the local interrupt pin that triggers an interrupt.

- 0: indicates edge sensitive,
- 1: indicates level sensitive.

Only the local interrupt pins can be programmed as edge or level triggered. Timer interrupts are always treated as edges.

MASK: [Bit 16]

- 0: enables injection of the interrupt,
- 1: masks injection of the interrupt.

Bits [31:17] Bits [31:17] are Reserved. Should be written 0.

6.4 Inter-Processor Interrupt Registers

A processor generates inter-processor interrupts by writing to the Interrupt Command Register in its 82489DX Local Unit. Conceptually, this can be thought of as the processor providing the interrupt's Redirection Table Entry on the fly. Not surprisingly, the layout of the Interrupt Command Register resembles that of an entry in the Redirection Table. Note that the format of this register allows a processor to generate any interrupt. A processor may use this to forward device interrupts originally accepted by it to other processors.

All fields of the Interrupt Command Register are read-write by software with the exception of the Delivery Status field which is read-only. Writing to the 32-bit word that contains the interrupt vector causes the interrupt message to be sent.

Vector: [Bits 7-0]

The vector identifies the interrupt being sent. If the Delivery Mode is "Remote Read", then the Vector field contains the address of the register to be read in the remote 82489DX's Local Unit. The addresses are listed in the section discussing 82489DX Local Unit register summary. For example, for ID register, remote read address of 02 should be specified in vector field.

Delivery Mode: [Bits 10-8]

The Delivery Mode is a 3-bit field that specifies how the 82489DX listed in the destination field (bits 63:32) should act upon reception of this signal. Note that certain Delivery Modes will only operate as intended when used in conjunction with a specific Trigger Mode. These restrictions are indicated for each Delivery Mode.

000: (Fixed) means deliver the signal on the INT pin of all processors listed in the destination. Trigger Mode for "fixed" Delivery Mode can be edge or level.

001: (Lowest Priority) means deliver the signal on the INT pin of the processor that is executing at the lower priority among all the processors listed in the specified destination; Trigger Mode for "lowest priority" Delivery Mode can be edge or level.

010: Intel Reserved. Should not be used.

011: (Remote Read) is a request to a remote 82489DX Local Unit to send the value of one of its registers over the ICC bus. The register is selected by providing its address in the Vector field. The register value is latched by the requesting 82489DX and stored in the Remote Register where it can be read by the local processor. A Delivery Mode of "Remote Read" requires an "edge" Trigger Mode.

Interrupt Command Register [31:0]

Bits [31:20]	Bits [19:18]	Bits [17:16]	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bits [10:8]	Bits [7:0]
--------------	--------------	--------------	--------	--------	--------	--------	--------	-------------	------------

100: (NMI) means deliver the signal on the NMI pin of all processors listed in the destination, vector information is ignored. A Delivery Mode equal to "NMI" requires a "level" Trigger Mode.

101: (Reset) means deliver the signal to all local units listed in the destination. The destination local unit will assert/deassert its PRST output pin. All addressed 82489DX Local Units will assume their reset state but preserve their ID. One side effect of an ICC bus message with Delivery Mode equal to "Reset" that results in a deassert of reset is that all Local Units (whether listed in the destination or not) will reset their lowest-priority tie breaker arbitration ID to their Local Unit ID (see the section on the ICC bus for details). A Delivery Mode of "Reset" requires a "level" Trigger Mode. "RESET" should not be used with "self" or "all incl self" Shorthand mode since it will leave the system in non-recoverable reset state. If "RESET" is used with "all excl self" mode software should make sure that only one CPU executes this instruction in a MP system.

110: Intel Reserved. Should not be used.

111: Intel Reserved. Should not be used

Destination Mode: [Bit 11]

This field determines the interpretation of the Destination field.

0: (Physical Mode): in Physical Mode, a destination 82489DX is identified by its Local Unit ID. Bits 56 through 63 (8 MSB of the destination field) specify the 8-bit 82489DX Local Unit ID.

1: (Logical Mode): in Logical Mode, destinations are identified by matching on Logical Destination under the control of the Destination Format Register in each Local 82489DX. The 32-bit Destination field is the logical destination.

Delivery Status: [Bit 12]

Delivery Status is a 1-bit field that contains the current status of the delivery of this interrupt. Two states are defined:

0: (Idle) means that there is currently no activity for this interrupt;

1: (Send Pending) indicates that the interrupt has been injected, but its delivery is temporarily held up by other recently injected interrupts that are in the process of being delivered;

Delivery Status is software read-only; software writes to this field (as part of a 32-bit word) do not affect this bit. Software can read to find out if the current interrupt has been sent, and the Interrupt Command Register is available to send the next interrupt. If the Interrupt Command Register is overwritten before the Delivery Status is "Idle", then the destiny of that interrupt is undefined; i.e., the interrupt may have been lost.

Bit 13: Bit 13 is Reserved. Should be written 0.

Level: [Bit 14]

Software can use this bit in conjunction with the Trigger Mode bit when issuing an inter-processor interrupt to simulate assertion/deassertion of level sensitive interrupts.

To assert: Trigger mode = 1 and Level = 1.

To deassert: Trigger mode = 1 and Level = 0.

For example, a message with Delivery Mode of "Reset", a Trigger Mode of "Level", and Level bit of 0 deasserts Reset to the processor of the addressed 82489 DX Local Unit(s). As a side effect, this will also cause all 82489DX to reset their Arbitration ID to their unit ID. (The Arb ID is used for tie breaking in lowest priority arbitration.)

Trigger Mode: [Bit 15]

Software can use this in conjunction with Level Assert/Deassert to generate interrupts that behave as edges or levels.

0: Edge

1: Level

Remote Read Status: [Bits 17,16]

This field indicates the status of the data contained in the Remote Read register. This field is read-only to software. Whenever software writes to the Interrupt Command Register using Delivery Mode "Remote Read" the Remote Read status becomes "in-progress" (waiting for the remote data to arrive). The remote 82489DX Local Unit is expected to respond in a fixed amount of ICC bus cycles. If the remote 82489DX Local Unit is unable to do so, then the Remote Read status becomes "Invalid". If successful, the Remote Read status resolves to "Valid". Software should poll this field to determine completion and success of the Remote Read command.

00: (invalid): the content of the Remote Read Register is invalid. This is the case after a Remote Read command issued and the remote 82489DX Local Unit was unable to deliver the Register content in time.

01: (in progress): a Remote Read command has been issued and this 82489DX is waiting for the data to arrive from the remote 82489DX Local Unit.

10: (valid): the most recent Remote Read command has completed and the remote register content in the Remote Read Register is valid.

11: reserved.

Destination Shorthand: [Bits 19,18]

This field indicates whether a shorthand notation is used to specify the destination of the interrupt and if so, which shorthand is used. Destination Shorthands do not use the 32-bit Destination field, and can be sent by software with a single 32-bit write to the 82489DX's Interrupt Command Register. Shorthands are defined for the following common cases: software self interrupt, interrupt to all processors in the system including the sender, interrupts to all processors in the system excluding the sender.

00: (dest field): means that no shorthand is used. The destination is specified in the 32-bit Destination field in the second word (bits 32 to 63) of the Interrupt Command Register.

01: (self): means that the current 82489DX Local Unit is the single destination of the interrupt. This is useful for software interrupts. The Destination field in the Interrupt Command Register is ignored. RESET Delivery mode should not be used with self destination. Only FIXED delivery mode should be used with SELF.

10: (all incl self): means that the interrupt is to be sent to "all" processors in the system including the processor sending the interrupt. The 82489DX will broadcast a message with destination unit ID field set to all ones. RESET assert Delivery mode should not be used with "all incl self" destination.

11: (all excl self): means that the interrupt is to be sent to "all" processors in the system with the exception of the processor sending the interrupt. The 82489DX will broadcast a message with destination unit ID field set to all ones. All-excl-self is useful during selection of a boot processor (init) and also for TLB flush where "self" is flushed using the processor flush instruction. Only one CPU in a MP system should execute "all excl self" destination if used with RESET Delivery mode.

Bits [31:20] Bits [31:20] are Reserved. They should be written 0.

Interrupt Command Register [63:32]

Bits [63:32]

Destination: [Bits 63-32]

This field is only used when the Destination Shorthand is set to "Dest Field". If Destination Mode is Physical Mode, **then the 8 MSB contain a Destination unit ID.** If Logical Mode, the full 32-bit Destination field contains the logical address. The enabling is done by Destination Format Register.

6.5 IRR, ISR, TMR Registers

INTERRUPT ACCEPTANCE

All 82489DX Local Units listen to all messages sent over the ICC bus. For each message, the local unit first checks if it belongs to the destination in the

ADVANCE INFORMATION

message. It does this by matching the 32-bit Destination field in the message against its logical Destination Register, if the message addresses in logical mode, and against its physical ID, if the message addresses in physical mode. All 82489DX Local Units that match are said to "belong to the group".

Each 82489DX Local Unit contains three 256-bit registers that play a role in the acceptance of interrupts and in dispensing accepted interrupts to the local processor. Each of these registers is a bit array where bit position *i* tracks information about the interrupt with vector *i*. These bits track information about the (PINT) maskable interrupts only. They are not relevant for NMI, RESET or ExtINT type of interrupts. The Interrupt Request Register (IRR) contains the interrupts accepted by this 82489DX Local Unit but not yet dispensed to the processor. The In Service Register (ISR) contains the interrupts that are currently in service by the processor, i.e., the interrupts that have been dispensed to the processor but for which the processor has not yet signaled the End-Of-Interrupt.

Note that the 82489DX's IRR and ISR registers have the same meaning and operation as in the 8259A in fully nested/non-specific EOI mode. Note also that these registers play no role in providing 8259A compatibility. Compatibility is handled by making an ex-

ternal 8259A-style controller directly visible to the processor and having the 82489DX become transparent.

Each interrupt has a vector associated with it, which determines the bit position, and hence the priority for the interrupt. When an interrupt is being serviced, all equal or lower priority interrupts are automatically masked by the 82489DX Local Unit.

The Trigger Mode Register (TMR) indicates for each interrupt whether the interrupt is edge or level. This information is transmitted with each 82489DX interrupt request message and reflects the Trigger Mode bit in the interrupt's Redirection Table entry. If an interrupt goes in service and the TMR bit is 0 (edge), then the interrupt's IRR bit is cleared at the same time the ISR bit is set. If the TMR bit is 1 (level), then the IRR bit is not cleared when the interrupt goes in service. In the latter case, the IRR bit mirrors the state of the interrupt's input pin.

The following diagram shows 82489DX operation with devices A and B sharing a level triggered interrupt input. The diagram illustrates how Remote IRR, and the IRR bit at the destination 82489DX track the state of INTIN. It also illustrates how an EOI is followed immediately by re-raising the interrupt as long as the INTIN is still asserted by some device.

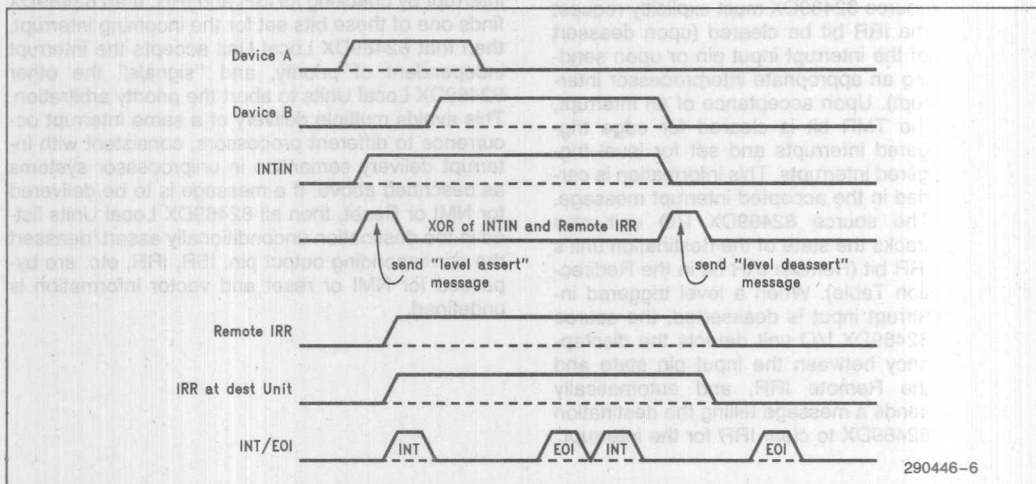


Figure 6. Interrupt Sharing

ISR, IRR, and TMR are read-only by software. Each of these 256-bit registers is accessed as four separate 32-bit registers. Note that there is no general Interrupt Mask Register (IMR) as in the 8259A. The processor masks interrupts temporarily by writing to the local unit's Task Priority Register (described shortly).

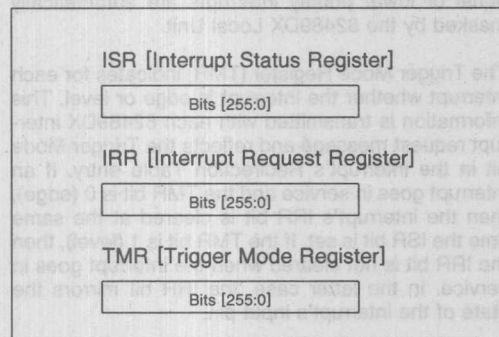


Figure 7. ISR, IRR, and TMR

TMR (Trigger Mode Register):

If 0 [edge triggered] the corresponding IRR bit is automatically cleared when interrupt service starts. If 1 [level triggered] this is not the case; instead, the source 82489DX must explicitly request the IRR bit be cleared (upon deassert of the interrupt input pin or upon sending an appropriate interprocessor interrupt). Upon acceptance of an interrupt, the TMR bit is cleared for edge triggered interrupts and set for level triggered interrupts. This information is carried in the accepted interrupt message. The source 82489DX I/O unit also tracks the state of the destination unit's IRR bit (Remote IRR bit in the Redirection Table). When a level triggered interrupt input is deasserted, the source 82489DX I/O unit detects the discrepancy between the input pin state and the Remote IRR, and automatically sends a message telling the destination 82489DX to clear IRR for the interrupt.

IRR (Interrupt Request Register):

It contains the active interrupt requests that have been accepted, but not yet dispensed by this 82489DX Local Unit. A bit in IRR is set when the 82489DX Local Unit accepts the interrupt. When TMR is 0, it is cleared when the interrupt is serviced; when TMR is 1, it is cleared when the 82489DX Local Unit receives a message to clear it.

ISR (In Service Register):

It marks the interrupts that have been delivered to the processor, but that have not been fully serviced in that an End-Of-Interrupt has not yet been received. The ISR register reflects the current state of the processor's interrupt stack.

ACCEPTANCE MECHANISM

Interrupt acceptance proceeds as follows. If the delivery mode is Fixed, then each unit in the destination group unconditionally accepts the interrupt message and sets the interrupt's IRR bit. If the delivery mode is Lowest Priority, then each processor in the group first checks if it is currently the focus of the interrupt by checking its ISR and IRR. If an 82489DX finds one of these bits set for the incoming interrupt, then that 82489DX Local Unit accepts the interrupt independent of priority, and "signals" the other 82489DX Local Units to abort the priority arbitration. This avoids multiple delivery of a same interrupt occurrence to different processors, consistent with interrupt delivery semantics in uniprocessor systems as described above. If a message is to be delivered for NMI or Reset, then all 82489DX Local Units listed in the destination unconditionally assert/deassert the corresponding output pin. ISR, IRR, etc. are bypassed for NMI or reset and vector information is undefined.

The acceptance decision process is illustrated in the flow chart below.

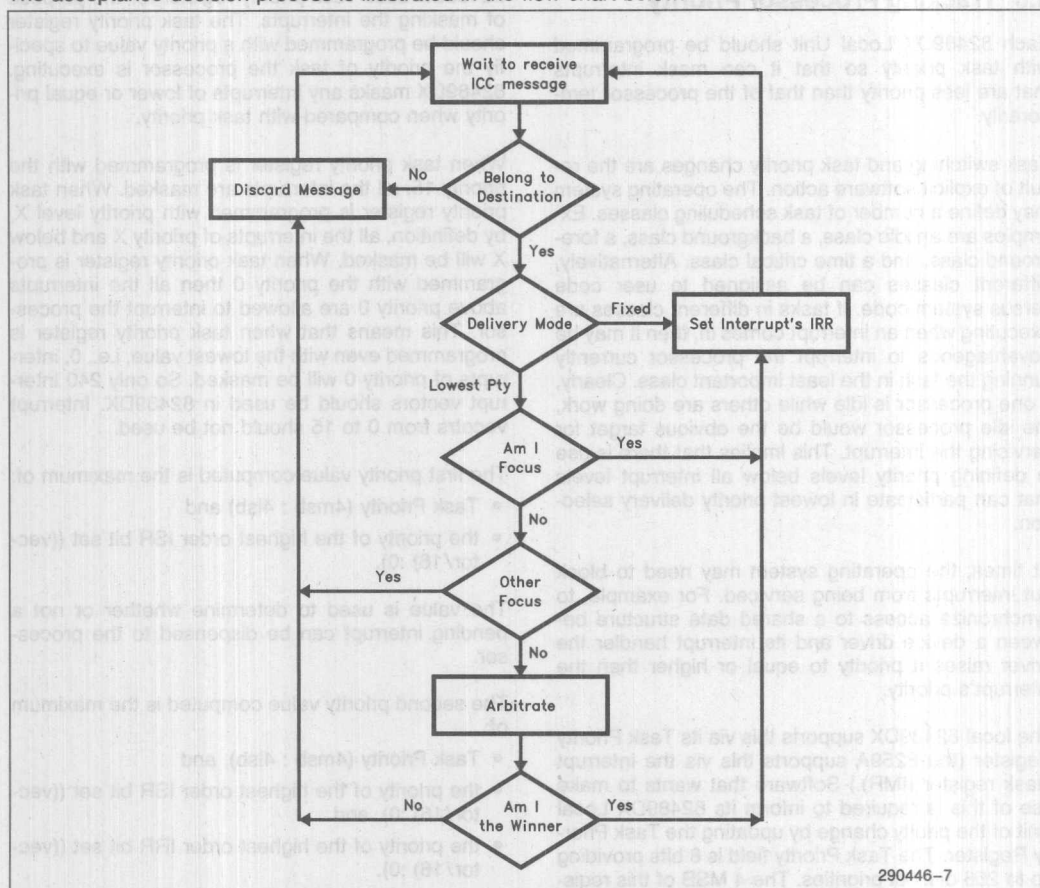


Figure 8. Interrupt Acceptance Flow Chart

Task Priority Register	Bit (7:0)
From the information in the Task Priority Register and the priority information derived from the IRR and IRR register, the 82489DX Local Unit computes two additional priority values:	
Bit [7:5] Bits [7:5] are Reserved. They should be written 0.	
Bit [7:0] Task Priority	
Bit [7:0] are used to specify the task priority.	

6.6 Tracking Processor Priority

Each 82489DX Local Unit should be programmed with task priority so that it can mask interrupts that are less priority than that of the processor temporarily.

Task switching and task priority changes are the result of explicit software action. The operating system may define a number of task scheduling classes. Examples are an idle class, a background class, a foreground class, and a time critical class. Alternatively, different classes can be assigned to user code versus system code. If tasks in different classes are executing when an interrupt comes in, then it may be advantageous to interrupt the processor currently running the task in the least important class. Clearly, if one processor is idle while others are doing work, the idle processor would be the obvious target for servicing the interrupt. This implies that there is use in defining priority levels below all interrupt levels that can participate in lowest priority delivery selection.

At times, the operating system may need to block out interrupts from being serviced. For example, to synchronize access to a shared data structure between a device driver and its interrupt handler the driver raises its priority to equal or higher than the interrupt's priority.

The local 82489DX supports this via its Task Priority Register (the 8259A supports this via the interrupt mask register (IMR).) Software that wants to make use of this is required to inform its 82489DX Local Unit of the priority change by updating the Task Priority Register. The Task Priority field is 8 bits providing up to 256 distinct priorities. The 4 MSB of this register correspond to the 16 interrupt priorities while the 4 LSB provide more precision. Priorities are best noted as x:y, where x is the value of the 4 MSB and y is the value of the 4 LSB. For example, Task Priority Register values 0:y with $0 < y < 15$ (and 0 in the 4 MSB) can be used to represent the priorities of the task scheduling classes described above ($y = 0$ for idle; $y = 1$ for background; etc.). Except for interrupts with vectors 0 through 15 (which are often predefined by the processor) which all have priority 0:0, the priorities of all other interrupts and their handlers is x:0 with $1 < x < 15$ and is above the base task priorities 0:y.

For example, interrupt vector 123 has priority 7:0 ($123/16 = 7$) and can be masked by any task that raises its priority to a value equal or higher than 7:0.

82489DX uses Task priority register for the purpose of masking the interrupts. The task priority register should be programmed with a priority value to specify the priority of task the processor is executing. 82489DX masks any interrupts of lower or equal priority when compared with task priority.

When task priority register is programmed with the priority 15, all the interrupts are masked. When task priority register is programmed with priority level X, by definition, all the interrupts of priority X and below X will be masked. When task priority register is programmed with the priority 0 then all the interrupts above priority 0 are allowed to interrupt the processor. This means that when task priority register is programmed even with the lowest value, i.e., 0, interrupts of priority 0 will be masked. So only 240 interrupt vectors should be used in 82489DX. Interrupt vectors from 0 to 15 should not be used.

The first priority value computed is the maximum of:

- Task Priority (4msb : 4lsb) and
- the priority of the highest order ISR bit set ((vector/16) :0).

The value is used to determine whether or not a pending interrupt can be dispensed to the processor.

The second priority value computed is the maximum of:

- Task Priority (4msb : 4lsb), and
- the priority of the highest order ISR bit set ((vector/16) :0), and
- the priority of the highest order IRR bit set ((vector/16) :0).

This value is used during arbitration as part of lowest-priority delivery.

Task Priority Register

Bits [31:8]	Bits [7:0]
-------------	------------

From the information in the Task Priority Register and the priority information derived from the ISR and IRR register, the 82489DX Local Unit computes two additional priority values:

Bits [31:8] Bits [31:8] are Reserved. They should be written 0.

Bits [7:0] Task Priority

Bits [7:0] are used to specify the task priority.

6.7 Dispensing Interrupts

DISPENSING INTERRUPTS TO THE LOCAL PROCESSOR

Once a 82489DX Local Unit accepts an interrupt, it guarantees delivery of the interrupt to its local processor. (This part of the 82489DX functions similarly to an 8259A.) Dispensing a maskable interrupt to the local processor begins when the Local Unit asserts the INT pin of its processor. If the processor has interrupts enabled, it will respond by issuing an INTA cycle. This causes the Local Unit to freeze its internal priority state and release the 8-bit vector of the highest priority interrupt on the data bus where it is read by the processor and used to find the handler's entry point. The INT/INTA protocol also causes the interrupt's ISR bit to be set. The corresponding bit in the IRR register is only cleared if the TMR register indicates it should do so (edge triggered interrupts), otherwise (level triggered interrupts), IRR is only cleared when the Interrupt Input Pin is deasserted.

6.8 Spurious Interrupt Vector Register

SPURIOUS INTERRUPT

Note that it can happen that a level-triggered interrupt is deasserted right before its INTA cycle. In that case, all IRR bits may be clear and the prioritizer may not find a vector to give to the processor. To satisfy the processor's demand for a vector, instead, the 82489DX will return a spurious interrupt vector instead.

A similar situation may occur when the processor raises its Task Priority at or above the level of the interrupt for which the Processor INT pin is currently being asserted. When the INTA cycle is issued, the interrupt that was to be dispensed has become masked (masked but remembered).

Dispensing the spurious interrupt vector does not affect the ISR register, so the handler for this vector should just return without EOI. If the vector is shared with a valid interrupt, then the handler can read the vector's bit in the ISR register to check if it is invoked for the valid interrupt (ISR bit set) or not (ISR bit clear). Given the range of 240 vectors, overloading the spurious interrupt with a valid interrupt is not expected to be common practice. The spurious interrupt vector to be used by a Local Unit is programmable via the Spurious Interrupt Vector Register.

UNIT ENABLE

It is possible that Local Units exist in the system that do not have a processor to which to dispense interrupts. The only danger this represents in the system

is that if any interrupt is broadcast to all processors using lowest priority delivery mode when all processors are at the lowest priority, there is a chance that a Local Unit without the processor may accept the interrupt if this Local Unit happens to have the lowest Arb ID at the time. To prevent this from happening, all Local Units initialize in the disabled state and must be explicitly enabled before they can either start accepting or transmitting messages from the ICC bus. A disabled 82489DX Local Unit only responds to messages with Delivery Modes set to "Reset". Reset deassert messages should be sent in Physical Destination mode using the target's Local Unit ID since the logical destination information in the local units is undefined (all zeroes) when the 82489DX comes out of Reset.

Bits [31:9]	Bit 8	Bits [7:0]
-------------	-------	------------

Figure 9. Spurious Interrupt Vector Register

Bits [31 .. 9] Reserved Bits. Should be written 0.

Unit Enable:[Bit 8]

0: When a 0 is written to this bit, this Local Unit gets disabled with regard to responding to messages sent as well as transmitting on the ICC bus. It only responds to messages with Delivery Mode set to "Reset". Reading a 0 at this bit indicates that the unit is disabled.

1: When a 1 is written to this bit, the current Local Unit is enabled for both transmitting and receiving unit messages. Reading a 1 at this bit indicates that the unit is enabled.

Spurious Vector: [Bits 7-0]

For future compatibility, bits [3-0] should be 1111.

6.9 End-of-Interrupt (EOI) Register

Before returning from the interrupt handler, software must issue an End-Of-Interrupt (EOI) command to the 82489DX Local Unit. The data written to EOI register is don't care. This tells the 82489DX to clear the highest priority bit in the ISR register since the interrupt is no longer in service. Upon EOI, 82489DX goes through prioritization returning to the next highest priority activity. This can be a previously interrupted handler (from ISR), a pending interrupt request (from IRR), or an interrupted task (from Task Priority).

Bits [31:0]

Figure 10. EOI Register

Bits [31:0]: are don't care.

2

6.10 Remote Read Register

Since all 82489DX Local Units would typically occupy the same address range, an 82489DX local unit's registers can only be accessed by the local processor. From a system debugging point of view, this would mean that a large amount of state would become inaccessible if its corresponding processor hangs for whatever reason. To assist in the debugging of MP systems, the 82489DX support a mechanism that provides read-only access to any register in any other 82489DX local unit in the system.

To read any register in a "remote" 82489DX Local Unit, the processor writes to the Interrupt Command Register specifying a Delivery Mode equal to "Remote Read". The remote 82489DX is specified in the Destination field of the Interrupt Command Register in the usual fashion. Debug software would make sure that this selects a single 82489DX only—for example by using the target's 82489DX Local Unit ID in physical destination mode. Since no vector is associated with remote register access, the Vector field in the Interrupt Command Register is used to select the individual remote 32-bit register to be read. The selector value corresponds to the address (offset) of the register in the local 82489DX's address space. Sending a "Remote Read" command results in sending a message on the ICC bus. The destination 82489DX responds by placing the 32-bit content of the selected register on the ICC bus. This value is read by sending the 82489DX and place it in the Remote Register where software can get at it using regular register access to its 82489DX Local Unit. The Remote Register is software read-only. The contents of the Remote Register is valid when the Delivery Status in the Interrupt Command Register has become "Idle" again.

Remote Read Register

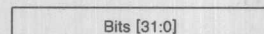


Figure 11. Remote Register

Bits [31:0] Bits [31:0] contain the contents of Remote Read Register.

6.11 82489DX Local Configuration

LOCAL VERSION REGISTER

Each 82489DX Local Unit contains a hardware Version Register that identifies this 82489DX Local Unit version. This register is read only.

Local Version Register

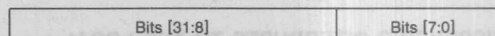


Figure 12. Local Version Register

Version: [Bits 7–0]

This is a version number that identifies this version. This field is hardwired and is read-only. Will be read as "1" for 82489DX.

Bits [31:8] Bits 31:8 are reserved.

6.12 82489DX Timer Registers

Overview

82489DX Local Unit contains one 32-bit wide programmable binary timer for use by the local processor. The timer can select its clock base from one of three possible clock inputs. A timer mode can be programmed to operate in either one-shot mode or periodic mode. The timer can be configured to interrupt the local processor with a vector.

Time Base

The 82489DX has two independent clock input pins:

1. The CLK pin provides the clock signal that drives the 82489DX's internal operation.
2. The TMBASE pin allows an independent clock signal to be connected to the 82489DX for use by the timer functions.

Signals from both CLK and TMBASE can be used as clock inputs that feed the timer. In addition, the 82489DX contains a divider that can be configured to divide either input clock signal. The divider can be programmed to divide the selected input clock by 2, 4, 8, or 16. CLK, TMBASE, and the output of the divider together provide three time bases: Base 0, Base 1, and Base 2. Base 0 is always equal to CLK; Base 1 is always equal to TMBASE; and base 2 is one of: CLK/2, CLK/4, CLK/8, CLK/16, TMBASE/2, TMBASE/4, TMBASE/8, or TMBASE/16. The timer can independently select one of these three time bases as its clock input as depicted in the following diagram.

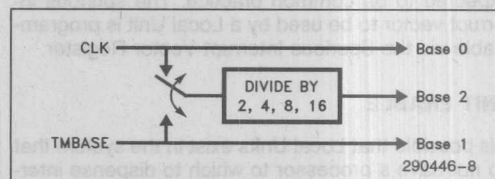


Figure 13. Time Bases

Bits [31:3]	Bit 2	Bits [1:0]
-------------	-------	------------

Figure 14. Divider Configuration Register

Bits [31:3] Bits 31 to 3 are reserved. They should be written 0.

Divider Input:

[Bit 2] Selects whether divider's input connects to the 82489DX Local Unit's CLK pin or TMBASE pin.

0: means the divider takes its input signal from CLK,

1: means use TMBASE.

Divide By: [Bits 1,0]

This field selects by how much the divider divides.

00: divide by 2

01: divide by 4

10: divide by 8

11: divide by 16

Timer

Software starts a timer going by programming its Initial Count Register. The timer copies this value into the Current Count Register and starts counting down at the rate of one count for each time base pulse. The time is one of Base 0, Base 1, or Base 2.

The timer has a programmable mode which can be One-Shot or Periodic. After the timer reaches zero in One-Shot mode, the timer simply stays at zero until it is reprogrammed. In Periodic mode, the timer automatically reloads its Current Count from the Initial Count and starts counting down again.

For the timer, interrupt generation can be disabled or enabled, and an arbitrary interrupt vector can be specified. When enabled and the timer reaches zero, an interrupt is generated at the 82489DX Local Unit. Timer generated interrupts are always treated as edges. They can only generate maskable interrupts to the local processor.

TIMER VECTOR TABLE

Bits [31:20]	Bits [19:18]	Bit 17	Bit 16	Bits [15:13]	Bit 12	Bits [11:8]	Bits [7:0]
--------------	--------------	--------	--------	--------------	--------	-------------	------------

Figure 16. Local Vector Table: Timer Entry

A timer set up with its interrupt masked is useful as a time base that can be sampled by the local processor by reading the Current Count Register, for the purpose of measuring the intervals. By mapping the 82489DX's register space into a read-only user page, safe and efficient performance monitoring of user programs can be supported.

If necessary, software may want to ensure that periodic timer interrupts on the different 82489DX Local Units are staggered such that the 82489DXs don't all deliver their interrupt (e.g., a timer slice interrupt) to their local processor at the same time. This staggering avoids bursts of contention for shared resources (bus, cache lines, dispatch queue, locks). Randomness occurring "naturally" may be sufficient to ensure staggering.

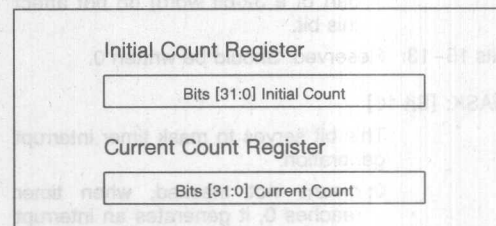


Figure 15. Initial Count and Current Count Registers

Initial Count: Software writes to this register to set the initial count for timer. This register can be written at any time. When written, its value is copied to the Current Count Register and countdown starts or continues from there. The Initial Count Register is read-write by software.

Current Count: This is the current count of timer. It is read-only by software and can be read at any time.

The timer is configured via its Local Vector Table entry shown below (see also Interrupt Control in this section).

Vector: [Bits 7-0]

This is the 8-bit interrupt vector to be used when timer generates an interrupt.

Bits 11–8: Reserved. Should be written 0.

Delivery Status: [Bit 12]

Delivery Status is a 1-bit field that contains the current status of the delivery of this interrupt. Two states are defined:

0: (Idle) means that there is currently no activity for this interrupt;

1: (Send Pending) indicates that the interrupt has been injected, but its delivery is temporarily held up by other recently injected interrupts that are in the process of being delivered; Delivery Status is software read-only; software writes to this field (as part of a 32-bit word) do not affect this bit.

Bits 15–13: Reserved. Should be written 0.

MASK: [Bit 16]

This bit serves to mask timer interrupt generation.

0: means not masked, when timer reaches 0, it generates an interrupt with vector at the 82489DX Local Unit

1: means masked, and no interrupt is generated.

Timer Mode: [Bits 17]

This field indicates the operation mode of timer.

0: (One-Shot): the Current Count Register remains at zero after the timer reaches zero, and software needs to reassign the timer's Initial Count Register to rearm the timer.

1: (Periodic): when the timer reaches zero, the Current Count Register is automatically reloaded with the value in the Initial Count Register, and the timer counts down again.

Timer Base: [Bits 19,18]

This field selects the time base input to be used by timer.

00: (Base 0) uses "CLKIN" as input;

01: (Base 1) uses "TMBASE";

10: (Base 2) uses the output of the divider (Base 2).

Bits [31:20] Bits [31:20] are Reserved. Should be written 0.

7.0 82489DX I/O UNIT REGISTERS

REGISTERS ADDRESSING SCHEME

The I/O Unit indirect addressing scheme uses two registers directly mapped into the processor's address space: the I/O Register Select register and the I/O Window register. The I/O register select register selects which I/O unit Register appears in the I/O Window register where it can be manipulated by software.

I/O Register Select Register

Bits [31:8]

Bits [7:0]

Figure 17. I/O Register Select Register

Bits [31:8]: Reserved. Should be written 0.

Bits [7:0]: I/O REGISTER SELECT: This register selects an 82489DX I/O unit register. The contents of the selected 32-bit register can be manipulated via the I/O Window Register. The I/O Register Select register is read-write by software.

I/O Window Register

Bits [31:0]

Figure 18. I/O Window Register

Bits [31:0] I/O WINDOW REGISTER: This register is mapped onto the I/O Unit's register selected by the I/O Register Select register. Readability/writability by software is determined by the I/O unit register that is currently selected.

The addresses (offsets to a platform-defined base address) of all registers are listed in the register summary section. Note that register offsets are aligned on 128-bit boundaries; in other words, registers are located only at every fourth 32-bit address. This eliminates the need for lane-steering glue logic when connecting the 82489DX's 32-bit data bus to a wider (64-bit and 128-bit) bus.

82489DX I/O UNIT CONFIGURATION

I/O Unit ID Register

Each 82489DX I/O Unit has a register that contains the I/O Unit's 8-bit ID. The I/O unit ID serves as a physical name of the 82489DX I/O Unit. It is used in arbitrating for ICC bus ownership when the I/O unit wants to access the ICC bus for sending any interrupt message. Unlike the local unit ID, the I/O unit ID is not latched-in from the address bus during hardware reset. The I/O unit ID is set to 0 during reset. The software has to write different ID into the I/O Units before starting interrupt messages on the ICC bus.

I/O Unit ID

Bits [31:24]	Bits [23:0]
--------------	-------------

Bits [31:24] I/O Unit ID:

The I/O unit ID serves as the physical "name" of the 82489DX unit used for arbitration purposes for the ICC bus usage. In a system with, say, four 82489DX, there are 4 Local Units and 4 I/O Units. All the 8 units should be assigned different ID. The IDs should start with 0 and each unit should have different ID.

Bits [23:0] Bits 23..0 are reserved. Should be written 0.

I/O Unit Version Register

Each 82489DX I/O Unit contains a hardware Version Register that identifies this 82489DX I/O unit version. This register is read only.

I/O Unit Version Register

Bits [31:24]	Bits [23:16]	Bits [15:8]	Bits [7:0]
--------------	--------------	-------------	------------

Version: [Bits 7-0]

This is a version number that identifies this version. This field is hardwired and is read-only. Will be read as "1" for 82489DX.

Bits [15:8] Bits [15:8] are reserved.

Redirection Table Entry

Bits [31:17]	Bit 16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bits [10:8]	Bits [7:0]
--------------	--------	--------	--------	--------	--------	--------	-------------	------------

Max Redir Entry: [Bits 23-16]

This is the entry number (0 being the lowest entry) of the highest entry in the Redirection Table. It is equal to the number of Interrupt Input Pins minus one of this I/O Unit. This field is hardwired and is read-only.

In the 82489DX I/O unit this is read as 15.

Bits [31:24] Bits [31:24] are reserved.

I/O UNIT INTERRUPT SOURCE REGISTERS

Redirection Tables

The Redirection Table has a dedicated entry for each interrupt input pin. Unlike IRQ pins of the 8259A, the notion of interrupt priority is completely unrelated to the position of the physical interrupt input pin on the 82489DX. Instead, software can decide for each pin individually what it wants the vector (and therefore the priority) of the corresponding interrupt to be. For each individual pin, the operating system can also specify whether the interrupt is signaled as edges or levels, as well as the destination and delivery mode of the interrupt. The information in the Redirection Table is used to translate the interrupt manifestation on the corresponding interrupt pin into an inter-82489DX message.

In order for a signal on an edge-sensitive Interrupt Input pin to be recognized as a valid edge (and not a glitch) the input level on the pin must remain asserted until the time 82489DX I/O Unit sends the corresponding message over the ICC bus. Only then will the source 82489DX be able to recognize a new edge on that Interrupt Input pin. That new edge will only result in a new invocation of the handler if its acceptance by the destination 82489DX causes the Interrupt Request Register bit to go from 0 to 1. (In other words, if the interrupt wasn't already pending at the destination.)

82489DX I/O unit has 16 Redirection Table entries. The layout of an entry in the Redirection Table is as follows:

Vector (Bits [7:0])

Interrupt vector for this interrupt

Delivery Mode (Bits [10:8])

000: Fixed

001: Lowest Priority

010: <reserved>

011: <reserved>

100: NMI

101: Reset

110: <reserved>

111: ExtINT

Destination Mode (Bit 11)

0: Physical

1: Logical

Delivery Status (Bit 12)

0: Idle

1: Send Pending

Bit 13

Bit 13: Reserved. Should be written 0.

Remote IRR (Bit 14)

Reflects the Remote IRR bit

0: Remote IRR bit is clear.

1: Remote IRR bit is set.

Trigger Mode (Bit 15)

0: Edge

1: Level

Mask (Bit 16)

0: Not Masked

1: Masked

Bits [31:17] Reserved. Should be written 0.

DESCRIPTIONS

Vector: [Bits 7–0]

The vector field is an 8-bit field containing the interrupt for this interrupt.

Delivery Mode: [Bits 10–8]

The Delivery Mode is a 3-bit field that specifies how the 82489DXs listed in the destination field should act upon reception of this signal. Note that remote read is not supported for I/O device interrupts. Note that certain Delivery Modes will only operate as intended when used in conjunction with a specific Trigger Mode. These restrictions are indicated for each Delivery Mode.

000: (Fixed) means deliver the signal on the INT pin of all processors listed in the destination. Trigger Mode for “fixed” Delivery Mode can be edge or level.

001: (Lowest Priority) means deliver the signal on the INT pin of the processor that is executing at the lower priority among all the processors listed in the specified destination; Trigger Mode for “lowest priority” Delivery Mode can be edge or level.

100: (NMI) means deliver the signal on the NMI pin of all processors listed in the destination, vector information is ignored. A Delivery Mode equal to “NMI” requires a “level” Trigger Mode.

101: (Reset) means deliver the signal to all processors listed in the destination by asserting/deasserting the 82489DX's Reset output pin. All addressed 82489DXs' Local Units will assume their reset state but preserve their unit ID. One side effect of a unit message with Delivery Mode equal to “Reset” that results in a deassert of reset is that all 82489DXs' Local Units (whether listed in the destination or not) will reset their lowest-priority tie breaker arbitration ID to their unit ID (see the section on the ICC bus for details). A Delivery Mode of “Reset” requires a “level” Trigger Mode.

111: (ExtINT) means deliver the signal to the INT pin of all processors listed in the destination as an interrupt that originated in an externally connected (8259A-compatible) interrupt controller. The Local Unit receiving this interrupt will activate ExtINTA in response to this interrupt message. A Delivery Mode of “ExtINT” requires an “edge” Trigger Mode. (See the section on Compatibility for details.)

Destination Mode [Bit 11]

This field determines the interpretation of the Destination field.

0: (Physical Mode): in Physical Mode, a destination 82489DX Local Unit is identified by its unit ID. Bits 56 through 63 (8 MSB of the destination field) specify the 8-bit unit ID.

1: (Logical Mode): in Logical Mode, destinations are identified by matching on Logical Destination under the control of the Destination Format Register in each 82489DX Local Unit. The 32-bit Destination field is the logical destination.

Delivery Status: [Bit 12]

Delivery Status is a 1-bit field that contains the current status of the delivery of this interrupt. Two states are defined:

0: (Idle) means that there is currently no activity for this interrupt;

1: (Send Pending) indicates that the interrupt has been injected, but its delivery is temporarily held up by other recently injected interrupts that are in the process of being delivered; Delivery Status is software read-only; software writes to this field (as part of a 32-bit word) do not affect this bit.

Bit 13: Bit 13 is Reserved. Should be written 0.

Remote IRR: [Bit 14]

This bit is used for level triggered interrupts; its meaning is undefined for edge-triggered interrupts. Remote IRR mirrors the interrupt's IRR bit of the destination 82489DX Local Unit. When the value of the bit disagrees with the state of the Interrupt Input line, a unit message is automatically sent to make the destination's IRR both reflect the new state of the Interrupt Input line, and then the Remote IRR bit is updated to track its associated IRR bit. Remote IRR is software read-only; software writes to this bit do not affect it.

Trigger Mode: [Bit 15]

The Trigger Mode field indicates the type of signal on the interrupt pin that triggers an interrupt.

0: indicates edge sensitive,

1: indicates level sensitive.

Mask: [Bit 16]

Use this bit to mask injection of this interrupt.

0: indicates that injection of this interrupt is not masked. An edge or level on an interrupt pin that is not masked results in the delivery of the interrupt to the destination.

1: indicates that injection of this interrupt is masked. Edge-sensitive interrupts signaled on a masked interrupt Input pin are simply ignored (i.e., it is not delivered and is not held pending). Level-asserts or deasserts occurring on a masked level-sensitive pin are also ignored and have no side effects. As expected, changing the mask bit from unmasked to masked while the level remains asserted has the side effect of deasserting the level. It is software's responsibility to deal with the case where the Mask bit is set after the interrupt message has been sent but before the interrupt is dispensed to the processor.

Bits [31:17] Bits [31:17] are reserved. Should be written 0.

Destination

Bits [63:32]

Destination: If the Destination Mode of this entry is "Physical Mode", then the 8 MSB [bits 56 through 63] contain an 82489DX Local Unit ID. If Logical Mode, then the Destination field potentially defines a set of processors. The interpretation of the 32-bit destination field is further enabled by the Destination Format Register in the 82489DX Local Units.

2

8.0 ICC BUS DEFINITION

Physical Characteristics

The ICC bus is a 5-wire synchronous bus connecting all 82489DXs (all I/O Units and all Local Units). Four of these five wires are used for data transmissions and arbitration, and one wire is a clock. The description refers to the logical state of the ICC bus. Electrical levels are just inverse of the logical state described. For example, the section describes that the ICC bus is 0000 when not transmitting any message. This refers to logical state. Electrically, the ICC bus is 1111 when not transmitting any message.

The bus is electrically an open-drain connection providing for both bus use arbitration and arbitration for lowest priority. Being open-drain, the bus is run at a "comfortable" speed such that design-specific termination tuning is not required. Furthermore, each 82489DX receiving a message or participating in an arbitration must be given enough time in a single bus cycle to latch the bus and perform some simple logic operations on the latched information in order to determine whether the next drive cycle must be inhibited.

Note that it is likely in MP systems that additional processors be located on plug-in boards. Since the ICC bus would be part of the connector, the 82489DX to ICC bus connection is defined so that it can be electrically isolated using external drivers. The 82489DX has separate ICC bus input and output pins that can be connected externally to the 82489DX to either provide or not provide isolation.

The isolation can also be used to provide a hierarchical connection of ICC buses electrically supporting large numbers of processors. The number of 82489DXs supported using the hierarchical connection is limited only by ICC bus bandwidth. It should be noted that ICC bus output low current is just 4 mA.

Bus Arbitration

Arbitration (both for use of the bus and for determining the lowest priority 82489DX) depends on all 82489DX message units operating synchronously. To deal with the event where multiple agents start transmitting simultaneously, a distributed arbitration approach is used. Bus arbitration uses a small number of arbitration cycles in the ICC bus. During

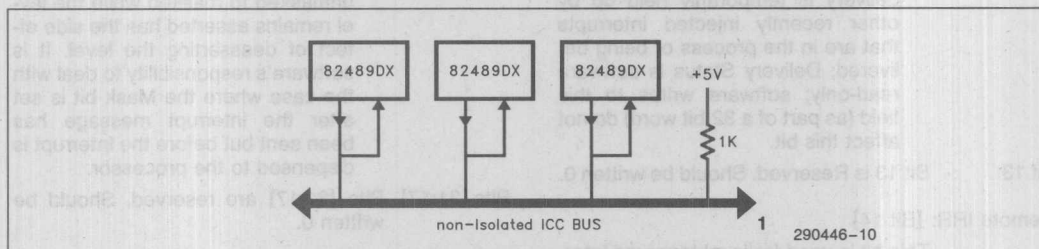


Figure 20. ICC Bus: Simple Direct Connection

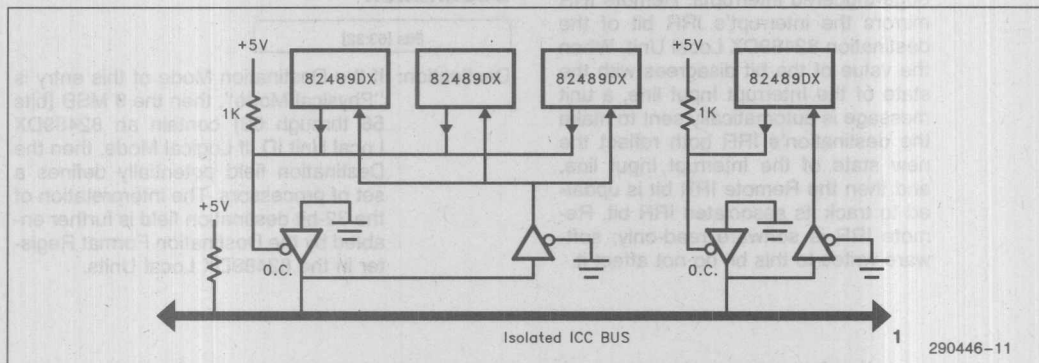


Figure 21. ICC Bits: Hierarchical Connection

these cycles, arbitration losers progressively drop off the bus until only the winner remains transmitting. The winner then transmits its actual inter-unit message. Once the sending of a message (including bus arbitration) has started, any possible contender must suppress transmission until enough cycles have elapsed for the message to be fully sent. The number of message cycles depends on the type of message being sent.

A bus arbitration cycle starts by the agent driving its unit ID on the ICC bus. High-order ID bits are driven first, successive cycles proceeding to the low bits of the ID. All losers in a given cycle drop off the bus, using every subsequent cycle as a tie breaker for the previous cycle. By the time all arbitration cycles are completed, there will be only a single agent left driving the bus.

The 8-bit unit ID (I7 I6 I5 I4 I3 I2 I1 I0) is chopped up in successive groups of 2 bits (I7 I6)(I5 I4)(I3 I2)(I1 I0). Each of these tuples is first decoded before driving them on the bus. The 0s and 1 indicate logical levels and not signal levels. The ICC bus is 0000 when not transmitting any message. The decoding used is:

ID Tuple		→	ICC Bus			
(I[i + 1])	I[i])		B3	B2	B1	B0
0	0	→	0	0	0	1
0	1	→	0	0	1	0
1	0	→	0	1	0	0
1	1	→	1	0	0	0

Note that the pattern generated on the ICC bus by tuple (I3 I2) will be represented as i32 i32 i32 i32. The lower case signifies this encoding.

Each tuple of the ID only contributes to a single wire, making it possible for an agent to determine with certainty whether to "drop off" or to continue arbitrating in the next cycle for the following two bits of the unit ID simply by checking whether the bus line the agent is driving is also the highest order 1 on the bus. Each ICC bus cycle therefore arbitrates 2 bits.

- 1: i76 i76 i76 i76 ICC bus arbitration
- 2: i54 i54 i54 i54
- 3: i32 i32 i32 i32
- 4: i10 i10 i10 i10
<message body>

Lowest-Priority Arbitration

Arbitration is also used to find the 82489DX Local Unit with the lowest processor priority. Lowest-priority arbitration uses the value of the 82489DX's Processor Priority value appended with an 8-bit Arbitration ID (Arb ID) to break ties in case there are multiple units executing at the lowest priority.

Using the constant 8-bit unit ID as the Arb ID has a tendency to skew symmetry since it would favor 82489DXs with low ID values. An 82489DX Local Unit's Arb ID is therefore not the unit ID itself but is derived from it. At reset, an 82489DX Local Unit's Arb ID is equal to its unit ID. Each time a message is broadcast over the ICC bus in lowest priority mode, all 82489DX Local Units increment their Arb ID by one, which gives them a different Arb ID value for the next arbitration. The Arb ID is then endian-reversed (LSB becomes MSB, etc.) to ensure better rotation of which 82489DX gets to have the lowest Arb ID next time around. The reversed Arb ID is then decoded to generate arbitration signals on the ICC bus as described above.

To support hot insertion of processor boards in a running MP system, a mechanism is provided to allow the 82489DX of the added processor to synchronize its Arb ID with the existing 82489DXs. This is accomplished by broadcasting a message with Delivery Mode equal to "Reset", Trigger Mode equal to "Level", and Level equal to 0. This message must be broadcast before the newly added 82489DX is allowed to participate in a lowest-priority arbitration. Depending on the exact sequence under which the newly inserted board is powered-up and initialized, this Arb ID synchronization may occur naturally if a Reset-deassert to the new 82489DX is part of that sequence. If not, the local processor can always send this as an inter processor interrupt (with a null destination), causing only the side effect of resetting all 82489DX Arb IDs.

ICC BUS MESSAGE FORMATS

The short message format is described first. Note that the first 19 cycles of both short and long message formats have the same interpretation.

- 1: i76 i76 i76 i76 ICC bus arbitration
- 2: i54 i54 i54 i54
- 3: i32 i32 i32 i32
- 4: i10 i10 i10 i10
- 5: DM M2 M1 M0 destination mode and delivery mode
- 6: "0" "0" L TM control bits
- 7: V7 V6 V5 V4 vector
- 8: V3 V2 V1 V0

9:	D31	D30	D29	D28	destination
10:	D27	D26	D25	D24	
11:	D23	D22	D21	D20	
12:	D19	D18	D17	D16	
13:	D15	D14	D13	D12	
14:	D11	D10	D09	D08	
15:	D07	D06	D05	D04	
16:	D03	D02	D01	D00	
17:	C	C	C	C	checksum for cycle 5 through 16
18:	"1"	"1"	"1"	"1"	postamble
19:	A	A	A	A	accept (1000 if OK, 1110 if preempt, else error)
20:	"0"	"0"	"0"	"0"	idle 1
21:	"0"	"0"	"0"	"0"	idle 2

Cycles 1 through 4 are bus arbitration as described earlier. Cycle 5 (DM M2 M1 M0) is the Destination Mode which is 0 for Physical mode and 1 for Logical Mode, and the Delivery Mode of the message. The encoding used for the Delivery Mode in the message is identical to the encoding used for the Delivery Mode in the Redirection Table, Local Vector Table, and Interrupt Command Register.

M2	M1	M0	Delivery Mode
0	0	0	Fixed
0	0	1	Lowest Priority
0	1	0	<reserved>
0	1	1	Remote Read
1	0	0	NMI
1	0	1	Reset
1	1	1	ExtINT

Cycle 6 contains the Control Bits of the message. The control bits are:

- TM (Trigger Mode): indicates whether this message corresponds to an edge or level;
- L (Level): indicates whether this is an Assert or a Deassert of a "level" signal. L is undefined when TM is edge.

6: "0" "0" L TM Control Bits

TM = Trigger Mode (0 = edge, 1 = level)

L = Level (0 = deassert, 1 = assert)

The length of the message is derived from the Delivery Mode, the Control Bits, and the Accept cycle of the message.

TM/L (AAAA)

	Edge	Level = Assert	Level = Deassert
Fixed	Short	Short	Short
Lowest Priority	Short (1110)	Short (1110)	Short (1110)
	Long (1000)	Long (1000)	Short
Remote Read	Long	Long	Short
NMI	Short	Short	Short
Reset	Short	Short	Short
ExtINTA	Short	Short	Short

Cycles 7 and 8 are the 8-bit interrupt vector. The vector is only defined for Delivery Modes Fixed, and Lowest-priority. For Delivery Mode of "Remote Read", the vector field contains the address of the register to be read remotely.

If DM is 0 (physical mode), then cycles 9 and 10 are the unit ID and cycles 11 through 16 are zero. If DM is 1 (logical mode), then cycles 9 through 16 are the 32-bit Destination field. The interpretation of the logical mode 32-bit Destination field is performed by the Local Units using the Destination Format Register. The sending 82489DX knows whether it should (incl) or should not (excl) respond to its own message.

Cycle 17 is a checksum over the data in cycles 5 through 16. The checksum is computed by adding all 4-bit quantities of cycles 5 through 16, feeding carry out of the MSB back into the LSB. This protects the data in these cycles against transmission errors. The (single) 82489DX driving the message provides this checksum in cycle 17.

Cycle 18 is a postamble cycle driven as 1111 by the sending 82489DX allowing all 82489DXs to perform various internal computations based on the information contained in the received message. One of the computations takes the computed checksum of the data received in cycles 5 through 16 and compares it against the value in cycle 17. If any 82489DX computes different checksum than the one passed in cycle 17, then that 82489DX will signal an error on the ICC bus in cycle 19 by driving it as 1111. If this happens, all 82489DXs will assume the message was never sent and the sender must try sending the message again, which includes re-arbitrating for the ICC bus. In lowest priority delivery when the interrupt has a focus processor, the focus 82489DX will signal this by driving 1110 during cycle 19. This tells all the other 82489DXs that the interrupt has been accepted, the 82489DXs is preempted, and short message format is used. All (non-focus) 82489DXs will drive 1000 in cycle 19. Under lowest priority mode, 1000 implies that the interrupt currently has no focus processor and that priority arbitration is required to complete the delivery. In that case, long message format is used. If cycle 19 is 1000 for non Lowest Priority mode, then the message has been accepted and is considered sent.

19:EEEE

1000	OK
1110	preempt
<others>	error (drive error as 1111)

When an 82489DX detects and reports an error during the error cycle, that 82489DX will simply listen to the bus until it encounters two consecutive idle (0000) cycles. These two idle cycles indicate that the message has passed and a new message may be started by anyone. This allows an 82489DX that got itself out of cycle on the ICC bus to get back in sync with the other 82489DXs.

Long Message Format

Cycles 1 through 19 of the long message format are identical to cycles 1 through 19 of the short message format. As mentioned, long message format is used in two cases:

- (1) Lowest Priority delivery when the interrupt does not have a focus. Cycles 20 through 27 are eight arbitration cycles where the destination 82489DXs determine the one 82489DX with lowest processor priority/ARB ID value.
- (2) Remote Read messages. Cycles 20 through 27 are the 32-bit content of the remotely read register. This information is driven on the bus by the remote 82489DX.

Cycle 28 is an Accept cycle. In lowest priority delivery, all 82489DXs that did not win the arbitration (including those that did not participate in the arbitration) drive cycle 28 with 1000 (co accept), while the winner 82489DX drives 1111. If cycle 28 reads 1111, then all 82489DXs know that the interrupt has been accepted and the message is considered delivered. If cycle 28 reads 1100 (or anything but 1111 for that matter), then all 82489DXs assume the message was unaccepted or an error occurred during arbitration. The message is considered undelivered, and the sending 82489DX will try delivering the message again.

For Remote Read messages, cycle 28 is driven as 1100 by all 8 2489DXs except the responding remote 82489DX, who drives the bus as 1111 in case it was able to successfully supply the requested data in cycles 20 through 27. If cycle 28 reads 1111 the data in cycles 20 through 27 is considered valid; otherwise, the data is considered invalid. The source 82489DX that issued the Remote Read uses cycle 28 to determine the state of the Remote Read Status field in the Interrupt Command Register (valid or invalid). In any case, a Remote Read request is always successful (although the data may be valid or invalid) in that a Remote Read is never retried. The reason for this is that Remote Read is a debug feature, and a "hung" remote 82489DX that is unable to respond should not cause the debugger to hang.

2

Cycles 29 and 30 are two idle cycles. The ICC bus is available for sending the next message at cycle 31. The two idle cycles at the end of both short and long messages, together with non zero (i.e., non idle) encoding for certain other bus cycles allow an ICC bus agent that happens to be out of phase by one cycle to sync back up in one message simply by waiting for two consecutive idle cycles after reporting its checksum error. This makes use of the fact that valid arbitration cycles are never 0000.

1:	i76	i76	i76	i76	ICC bus arbitration
2:	i54	i54	i54	i54	
3:	i32	i32	i32	i32	
4:	i10	i10	i10	i10	
5:	DM	M2	M1	M0	delivery mode
6:	"0"	"0"	L	TM	control bits
7:	V7	V6	V5	V4	vector
8:	V3	V2	V1	V0	
9:	D31	D30	D29	D28	destination
10:	D27	D26	D25	D24	
11:	D23	D22	D21	D20	
12:	D19	D18	D17	D16	
13:	D15	D14	D13	D12	
14:	D11	D10	D09	D08	
15:	D07	D06	D05	D04	
16:	D03	D02	D01	D00	
17:	C	C	C	C	checksum for cycles 5 through 16
18:	"1"	"1"	"1"	"1"	postamble
19:	A	A	A	A	accept (1000 if OK, 1110 if preempt, else error)
20:	p76	p76	p76	p76	lowest priority arbitration or 32 bits of remote register processor priority
21:	p54	p54	p54	p54	
22:	p32	p32	p32	p32	
23:	p10	p10	p10	p10	
24:	a76	a76	a76	a76	
25:	a54	a54	a54	a54	arbitration ID
26:	a32	a32	a32	a32	
27:	a10	a10	a10	a10	
28:	A	A	A	A	accept
29:	"00"	"0"	"0"	"0"	idle1
30:	"0"	"0"	"0"	"0"	idle2

9.0 HARDWARE TIMINGS

This section covers the following:

— Timing Diagram Notation

— 82489DX Register Access Timing Diagrams with Descriptions

A block diagram of the configuration of the CPU module of a MP system is shown. This in no way is intended to be a complete representation of 486/Intel Cache/Intel Cache Controller connections. It is intended to show all the 82489DX connections, and how they connect to other components on and off the module. This module has arbitrarily been drawn with a 64-bit data bus to show how the expanded address space architecture fits. The unit can be similarly attached to either a 32-bit or 128-bit data bus, with total transparency to shrink-wrap software.

In this configuration, the 82489DX uses the same clock source as the processor and cache. However, it is quite possible to consider 82489DX as a memory bus device and hence supply 82489DX with the memory bus clock, which can be slower than the CPU module clock frequency.

In the configuration shown, the processor's INT and NMI pins could be supplied by other source to allow for the possibility that 82489DX can be totally bypassed if desired, by allowing those signals to be driven from off the module while the 82489DX is disabled. The reset signal generated by the 82489DX goes to the MBC (memory bus controller) which is required to drive configuration lines at reset time. This would probably be configured as a "warm" reset by the MBC.

A future version of cache controller may generate the chip select for 82489DX at a fixed memory location of hexFEE00000. By having the cache controller to provide the chip select signal, it would encourage a standard mapping for 82489DX address space. In some MBC designs, this signal should be connected to the MBC since 82489DX cycles limit bus pipelining by constraining how soon the next bus cycle can come. The 82489DX chip select can be generated by the MBC completely.

The address, data and most of the bus control signals share the respective bus with cache and cache controller. The block diagram shows attachment for only 6 address lines: A4–A9. A10 should be 0. This is all the 82489DX needs for operation, however, if the address lines are used to initialize 82489DX local ID at reset time, 8 address lines are required, A3–A10.

INTERFACING TO THE ICC BUS

The 82489DX has separate ICC bus input and output pins to facilitate using external drivers. The ICC bus input pins (MBI0-3) are TTL-level compatible CMOS inputs. The output pins (MBO0-3) are open-drain pins which required external pull-ups. The open-drain output buffers are small buffers with:

Sink current of < 4 mA. Special consideration must be exercised when driving large capacitive loads or long transmission lines. The pull-up resistor and the capacitive load constitute RC time constant that will affect the output transition times. This in turn will limit the operating frequency of the ICC bus.

When designing in the ICC bus, one needs to consider the loads that each 82489DX will be driving and whether external drivers should be used. In most situations, the ICC bus driven high (MBO pins pulled high by the external pull-up resistors) poses the most challenge. Simulating the target design on an electrical simulator (such as SPICE) will help greatly, as shown in the following examples.

First Order Buffer Models

Figure 21a and 21b are first order input buffer and output buffer models of the MBI and MBO pins. The open-drain of the MBO is modeled as a switch as the primary interest here is the MBO pins going high. These models can be used on SPICE simulations to

obtain first order behaviors. The parameters for these models are as follows:

- C_p (package capacitance) = 3 pF
- L_p (package inductance) = 15 nH
- R_b (bond wire resistance) = 0.08 Ω
- C_i (input buffer capacitance) = 3 pF
- C_o (output buffer capacitance) = 6 pF
- R_o (output buffer impedance) = 30 Ω –80 Ω

MBO Pull-up Resistor

To minimize the RC time constant, one would like to use the smallest pull-up resistor value possible. The MBO pins has a worst case lol-spec of 4 mA and $V_{CC} = 4.75$ V. This translates to a minimum pull-up of about 1 K Ω . Where stronger drive is needed (smaller pull-up resistors), external drivers must be used.

Driving Lumped Capacitance

In systems where external drivers are not used, the MBI pins will be tied to the MBO pins. Figure 21d is a SPICE simulation of the MBO output with a 1 K Ω pull-up driving lumped capacitive loads from 10 pF to 150 pF.

At a load of 50 pF, it takes about 30 ns to charge up to 2V. At 100 pF, it takes an additional 25 ns. Figure 21d can be used to estimate the loading delay at different lumped capacitive loads.

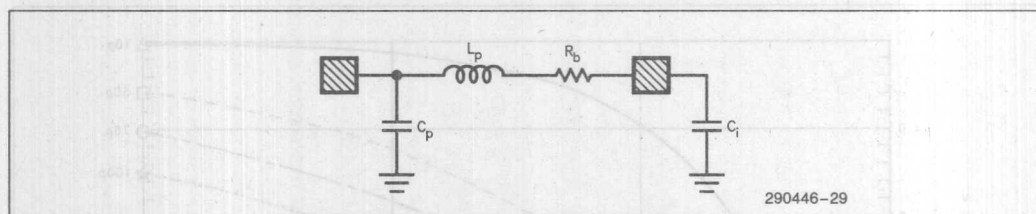


Figure 21a. First Order Input Buffer for MBI Pins

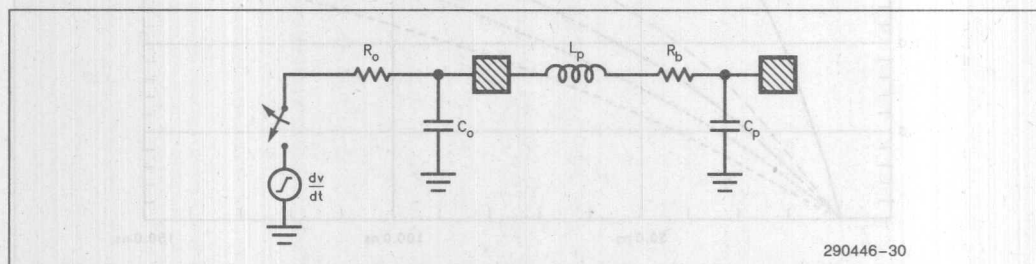


Figure 21b. First Order Open-Drain Output Buffer for MBO Pins

In real systems, the loads are made up of lumped capacitance and transmission lines. More accurate results can be obtained using transmission line models.

Driving Transmission Lines

Two device model

In this example the ICC bus is a signal line on an FR-4 printed circuit board. The line width is 6 mils. Line length of 12 inches and 18 inches are modeled. The FR-4 PC board has the following characteristics:

- resistivity = 0.6 m Ω /sq. (0.1 Ω /inch for 6 mil width)
- inductance = 60 pH/sq. (10 nH/inch for 6 mil width)
- capacitance = 0.55 nF/sq. in. (3.3 pF/inch for 6 mil width)

The ICC bus is shared by two 82489DXs, one at each end. The ICC bus is modeled as a transmission line. For the simulation, only one of the 82489DX is driving. A pull-up resistor of 2 K Ω is used at each end (1 K Ω equivalent value) as shown in Figure 21e. Figure 21f shows the signals at each end of the 12 inch transmission line. Trace 1 is the wave form at the driven end and trace 2 is the signal at the receiving end of the line. The 2 ns delay between the two signals is the propagation delay (or flight time) through the 12 inch transmission line. It takes about 35 ns for the voltage to charge up to 2V.

Figure 21g shows the received signal with different line length and with additional lumped capacitance. Trace 1 is for 12 inch only. Trace 2 is for 12 inch with additional 20 pF lumped capacitance to represent interconnect socket capacitance. Trace 3 is for 18 inch plus 20 pF. The presence of the 20 pF at each end of the 12-inch transmission line increases the delay time by 20 ns at 2V.

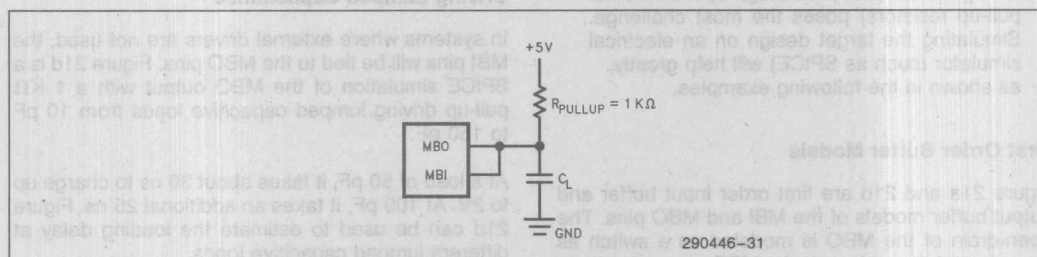


Figure 21c. ICC Bus Driving Lumped Capacitance

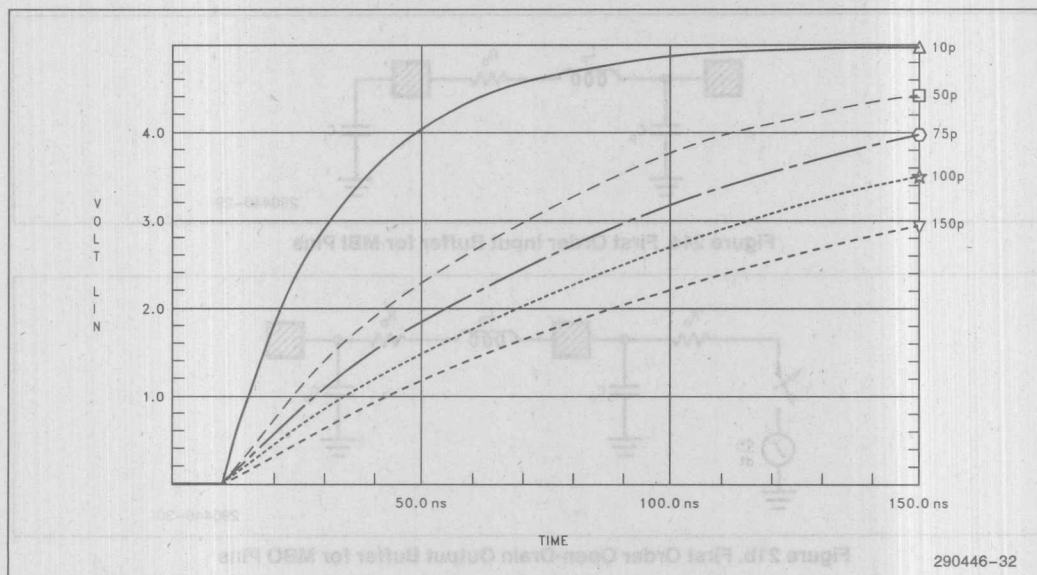


Figure 21d. 1 K Ω Pull-Up Driving Lumped Capacitance

Four Device Model

In this example (Figure 21h), the ICC bus is a 12-inch transmission line with four 82489DXs connected at 4 inch intervals. The loading at each junction consisted of the MBI and MBO buffers and a 20 pF lumped capacitance. 2 K Ω pull-ups are at each end of the transmission line.

As shown in Figure 21i, it takes more than 90 ns for the signal level at both ends to reach 2V.

One way to improve the low to high transition time is to use a stronger pull-up (smaller resistor value) which is possible using external line drivers with their larger current drive capabilities.

Figure 21j shows the difference in output when the model is used with 300 Ω pull-ups at each end of the transmission line.

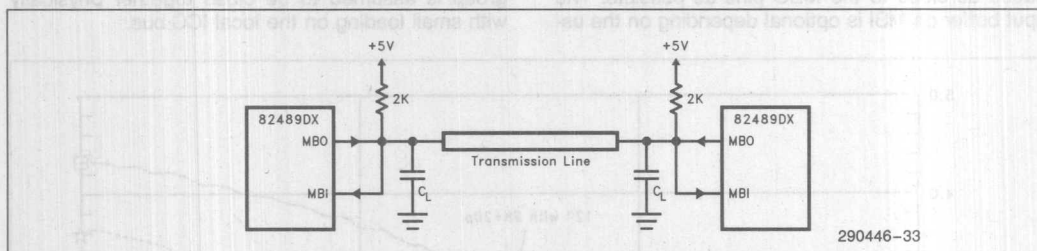


Figure 21e. Unbuffered ICC Bus with Two 82489DX

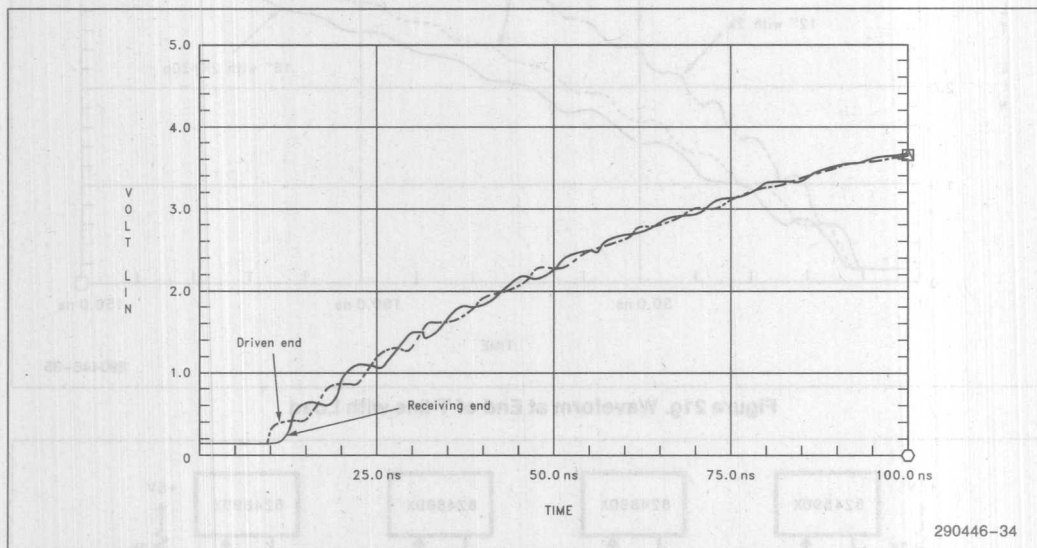


Figure 21f. Waveform at Both Ends of 12" T-line

External Drivers/Buffered ICC bus

The 82489DX has separate ICC Bus input (MBI) and output (MBO) pins that can be connected to external line drivers in systems that has appreciable loading on the ICC Bus or where modularity of the bus is needed.

Figure 21k is a typical implementation using external drivers with tri-state outputs. Drivers such as 74F125 or its equivalent can be used. The drivers should be placed as close to the MBO pins as possible. The input buffer on MBI is optional depending on the us-

ers ICC Bus scheme. The total delay through the drivers, buffers, transmission line, clock skews etc. must be calculated to ensure that all the ICC bus timing requirements are met.

A hierarchical bus connection can also be used in applications that cannot afford driver/buffer per unit and where bus loading are localized in cluster groups. Figure 21l shows such a connection where each cluster group is connected directly and drivers are used to connect to other clusters. Each cluster group is assumed to be close together physically with small loading on the local ICC bus.

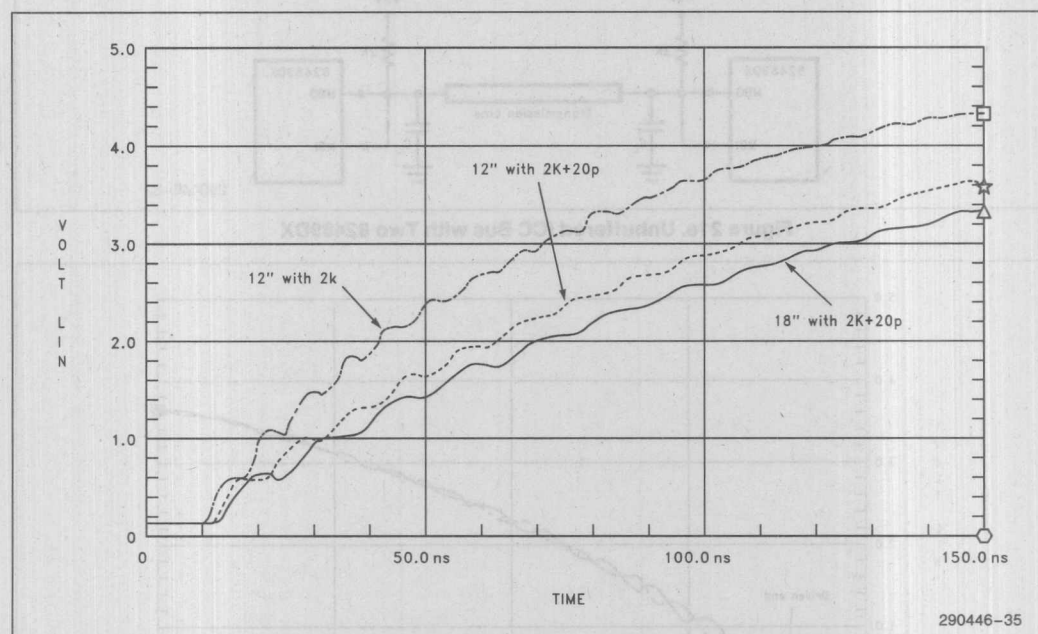


Figure 21g. Waveform at End of T-line with Load

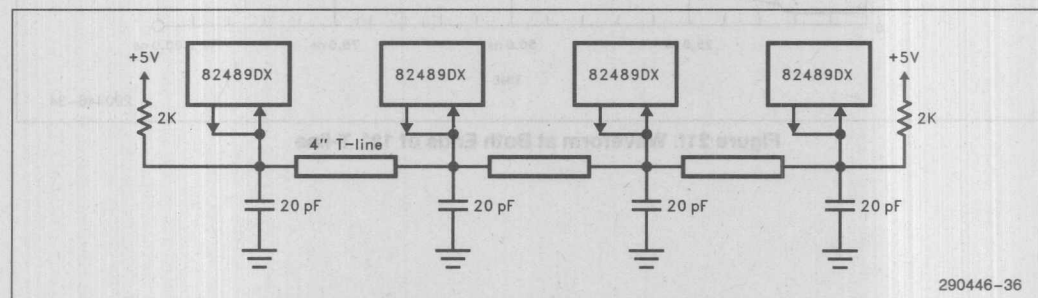


Figure 21h. Four 82489DX Configuration

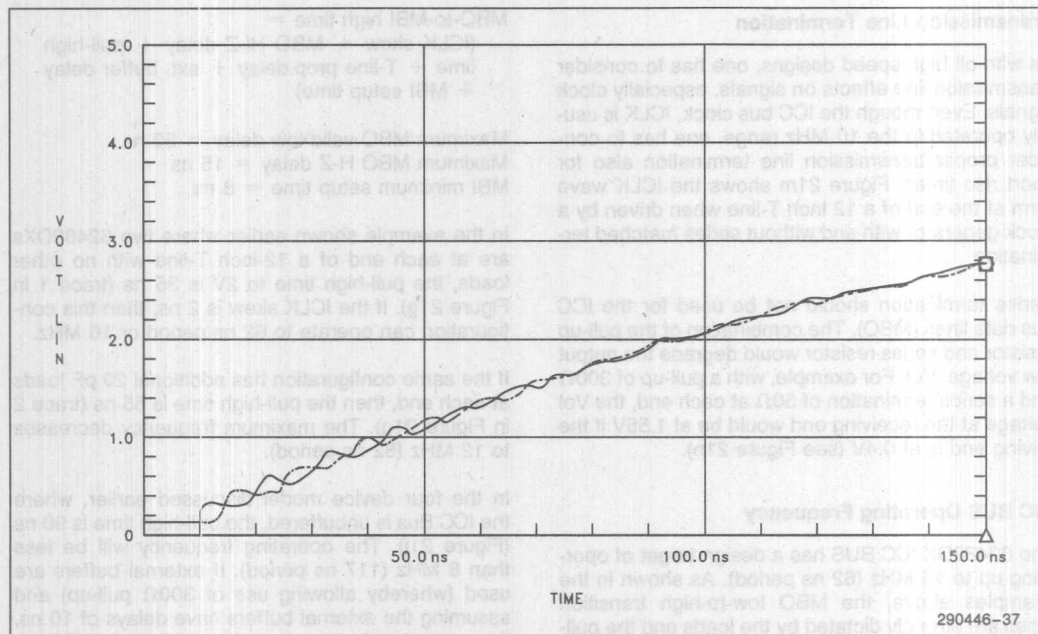


Figure 21i. Waveform for Four Devices on 12" T-line

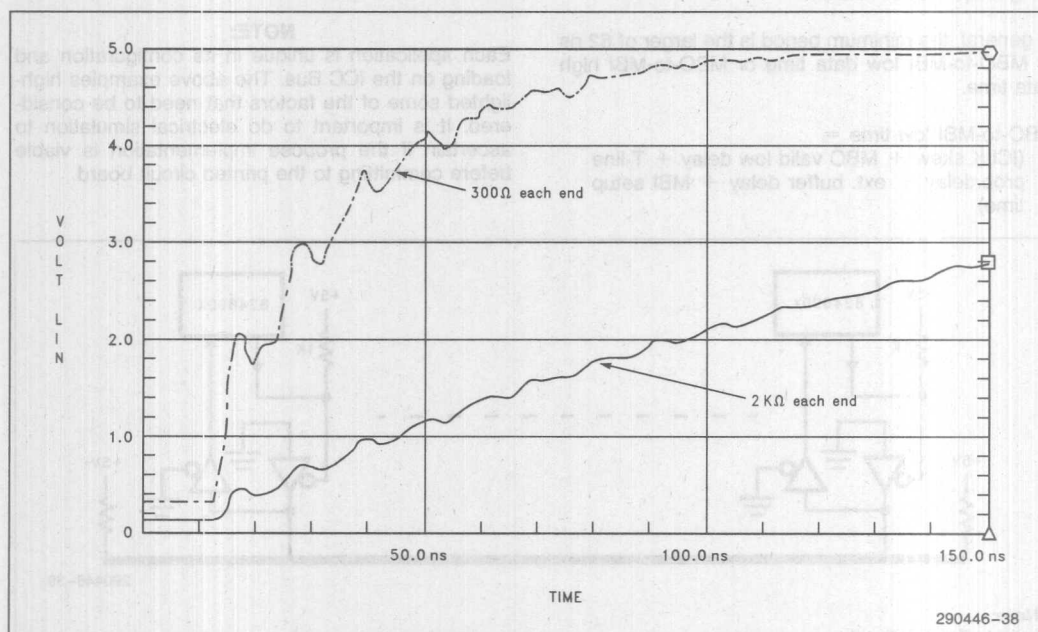


Figure 21j. Waveform with Different Pull-Ups

Transmission Line Termination

As with all high speed designs, one has to consider transmission line effects on signals, especially clock signals. Even though the ICC bus clock, ICLK is usually operated in the 10 MHz range, one has to consider proper transmission line termination also for short rise times. Figure 21m shows the ICLK wave form at the end of a 12 inch T-line when driven by a clock generator with and without series matched termination.

Series termination should not be used for the ICC bus data lines (MBO). The combination of the pull-up resistor and series resistor would degrade the output low voltage, Vol. For example, with a pull-up of 300Ω and a series termination of 50Ω at each end, the Vol voltage at the receiving end would be at 1.55V if the driving end is at 0.4V (see Figure 21n).

ICC BUS Operating Frequency

The 82489DX ICC BUS has a design target of operating up to 16 MHz (62 ns period). As shown in the examples above, the MBO low-to-high transition times are strongly dictated by the loads and the pull-ups used. This will in turn affect the maximum operating frequency of ICLK.

In general, the minimum period is the larger of 62 ns or MBO-to-MBI low data time or MBO-to-MBI high data time.

MBO-to-MBI low time =
(ICLK skew + MBO valid low delay + T-line prop.delay + ext. buffer delay + MBI setup time)

MBO-to-MBI high time =
(ICLK skew + MBO Hi-Z delay + pull-high time + T-line prop.delay + ext. buffer delay + MBI setup time)

Maximum MBO valid low delay = 50 ns

Maximum MBO H-Z delay = 15 ns

MBI minimum setup time = 8 ns

In the example shown earlier where two 82489DXs are at each end of a 12-inch T-line with no other loads, the pull-high time to 2V is 35 ns (trace 1 in Figure 21g). If the ICLK skew is 2 ns, then this configuration can operate to 62 ns period or 16 MHz.

If the same configuration has additional 20 pF loads at each end, then the pull-high time is 55 ns (trace 2 in Figure 21g). The maximum frequency decreases to 12 MHz (82 ns period).

In the four device model discussed earlier, where the ICC Bus is unbuffered, the pull-high time is 90 ns (Figure 21j). The operating frequency will be less than 8 MHz (117 ns period). If external buffers are used (whereby allowing use of 300Ω pull-up) and assuming the external buffers have delays of 10 ns, the operating frequency is limited by the MBO-to-MBI low time of 72 ns or 14 MHz.

NOTE:

Each application is unique in its configuration and loading on the ICC Bus. The above examples highlighted some of the factors that need to be considered. It is important to do electrical simulation to ascertain if the propose implementation is viable before committing to the printed circuit board.

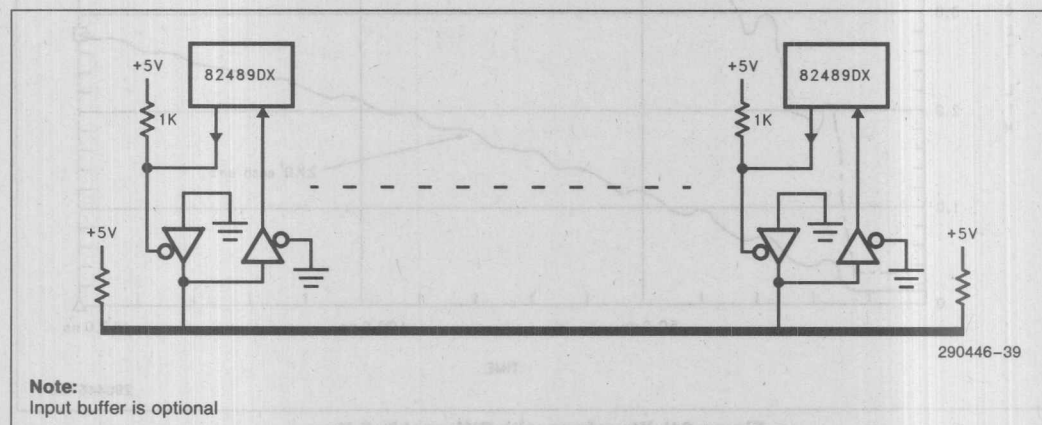


Figure 21k. External Driver/Buffer Implementation

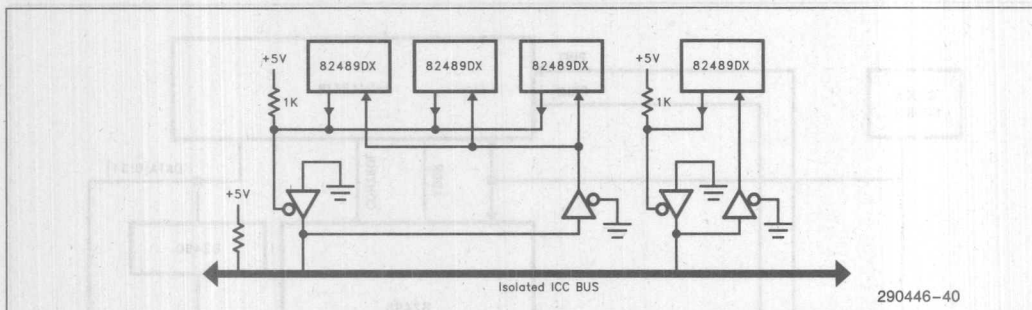


Figure 21l. ICC Bus: Hierarchical Implementation

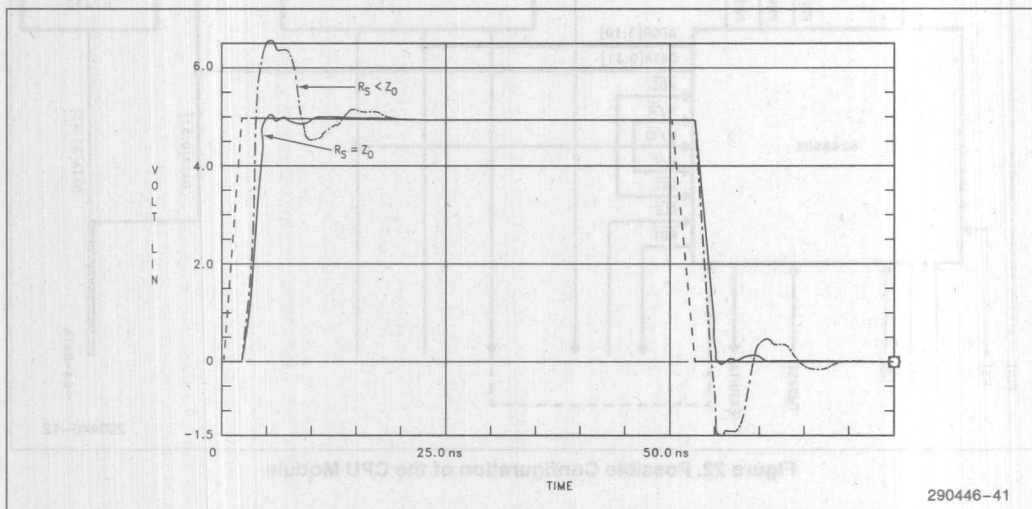


Figure 21m. ICLK Waveform on 12" T-line

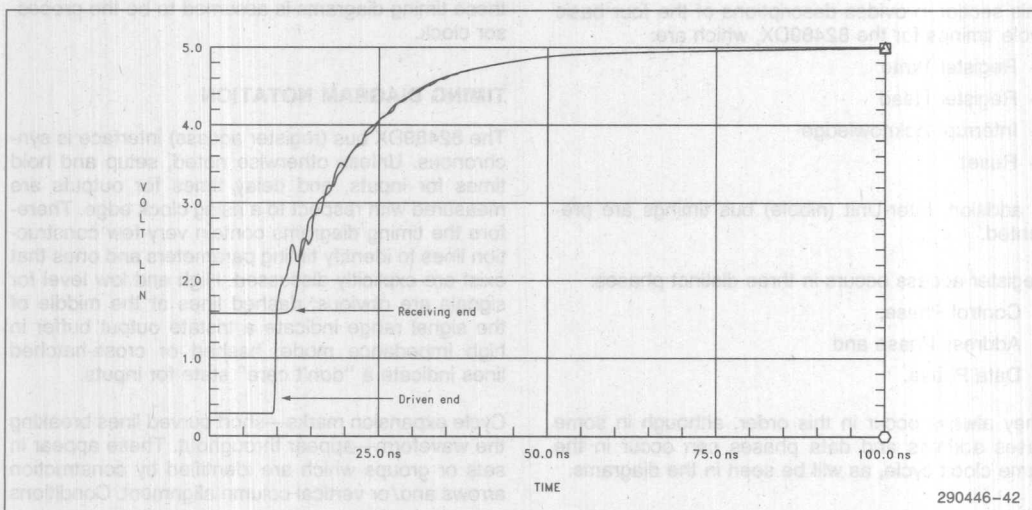


Figure 21n. Effect of Series Termination on MBO VOL

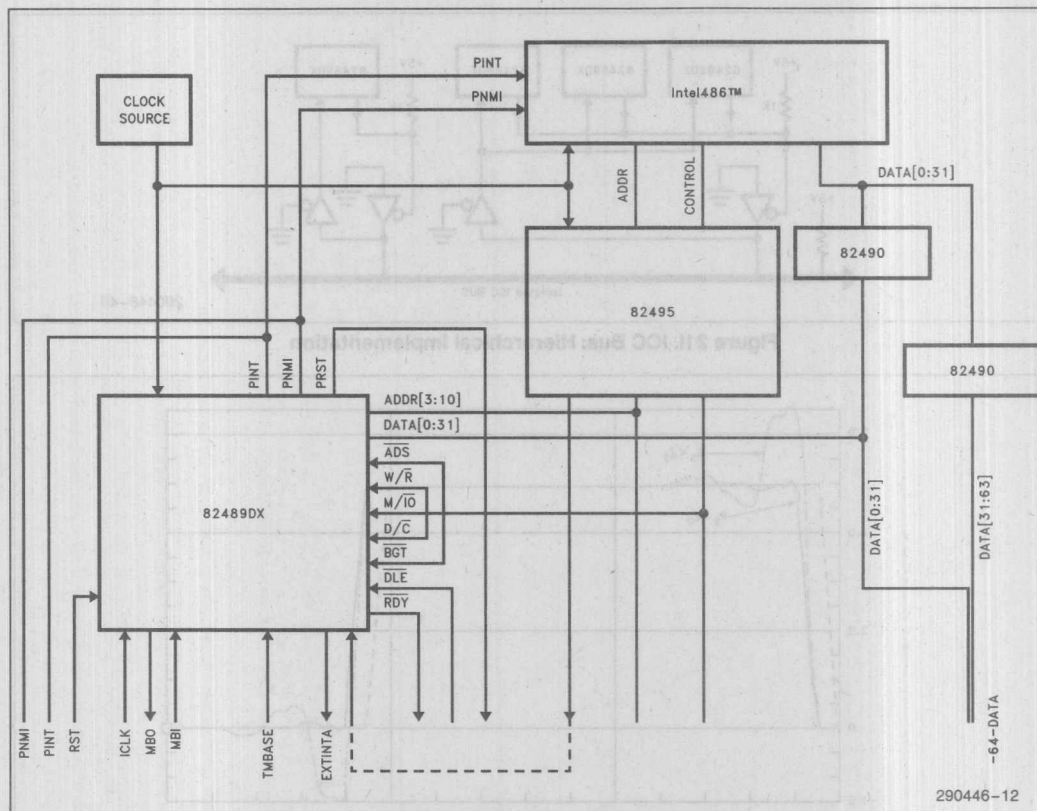


Figure 22. Possible Configuration of the CPU Module

9.1 82489DX Register Access Timing

This section provides descriptions of the four basic cycle timings for the 82489DX, which are:

- Register Write
- Register Read
- Interrupt Acknowledge
- Reset

In addition, Inter-Unit (nibble) bus timings are presented.

Register access occurs in three distinct phases:

1. Control Phase,
2. Address Phase and
3. Data Phase.

They always occur in this order, although in some cases address and data phases can occur in the same clock cycle, as will be seen in the diagrams.

NOTE:

As mentioned previously, the clock signal in all these timing diagrams is assumed to be the processor clock.

TIMING DIAGRAM NOTATION

The 82489DX bus (register access) interface is synchronous. Unless otherwise noted, setup and hold times for inputs, and delay times for outputs are measured with respect to a rising clock edge. Therefore the timing diagrams contain very few construction lines to identify timing parameters and ones that exist are explicitly discussed. High and low level for signals are obvious; dashed lines at the middle of the signal range indicate a tristate output buffer in high impedance mode; hashed or cross-hatched lines indicate a "don't care" state for inputs.

Cycle expansion marks—short curved lines breaking the waveform—appear throughout. These appear in sets or groups which are identified by construction arrows and/or vertical column alignment. Conditions

resulting in cycle expansion are listed at the bottom of each diagram, and the associated set of expansion marks indicates which signals must be stretched for that condition. Signals not so indicated are not affected by that condition, and continue without any cycle stretching. For example, the data phase of a transaction may be delayed, stretching all data related signals, while address related signals can continue to the next cycle.

Sample points for input signals are marked with bold, downward pointing arrows. Since sample timing for address, data and cycle definition signals is dependent on the timing of related control signals, a bar is used on top of the arrow to indicate the signal with the independent timing. Each signal group has exactly one independent signal. In general, independent signals are sampled on every clock, and therefore must meet setup and hold times on every clock edge. Signals having dependent timing, (indicated by the arrow with no bar), are only sampled when the associated independent signal is active, and therefore setup and hold times for dependent signals need only be met at the indicated sample points.

REGISTER WRITE TIMING

For discussion of this bus cycle, refer to Figure 23, 82489DX Timing Diagram 1. This shows the relationship between the three phases of the bus cycle.

The control phase is independently timed by the \overline{ADS} signal. The cycle definition signals [$\overline{M/\overline{IO}}$, $\overline{D/\overline{C}}$], are dependently sampled with \overline{ADS} as indicated by the bold sample point arrows labeled "C". The cycle definition signals will be sampled in the first clock when \overline{ADS} is active (low). The control signal should remain stable until \overline{ADS} goes inactive. For any valid 82489DX cycle, the memory bus controller should ensure that the \overline{ADS} pulse for a subsequent bus cycle is NOT presented until after the 82489DX asserts its \overline{RDY} pin (low) as shown.

The address phase is independently timed by the (\overline{BGT}) signal, as indicated by the bold sample point arrows labeled "A". This signal is actually used an address latch enable, however, its name is intended to imply that in most cases it can be directly driven by the Intel cache controller signal of the same name. The \overline{BGT} pulse may be delayed until the address bus is available, in which case all address and data phase signals will be stretched. Note that \overline{DLE} must not occur before \overline{BGT} . 82489DX does not start the internal cycle until \overline{BGT} is recognized with the appropriate chip select signal. If multiple \overline{ADS} has been issued without \overline{BGT} and a valid chip se-

lect, the internal cycle starts with the most recent \overline{ADS} cycle definition preceding the \overline{BGT} with valid chip select.

NOTE C:

Address information, including chip select (\overline{CS}), is sampled in the first clock when \overline{BGT} is active (low), and they must remain stable until \overline{BGT} goes inactive, OR until \overline{RDY} is asserted (low), whichever occurs first. \overline{CS} should be stable when \overline{ADS} is active.

In some configurations, \overline{BGT} may not be provided, and can be permanently tied low. In this case, the independent address timing will occur exactly one clock after the \overline{ADS} signal is first sampled low, and the dependent address information (address and chip select) will be assumed stable at this time. 82489DX recognize that independent \overline{BGT} timing is not provided by sampling a low state of \overline{BGT} at the time \overline{ADS} is first sampled low.

The data phase is independently timed by the \overline{DLE} signal, as indicated by the bold sample point arrows labeled "D". In the case of register writes, this signal work logically like a synchronous data latch enable. The \overline{DLE} pulse may be delayed until the data bus is available, in which case data and \overline{RDY} will be stretched.

NOTE D:

Write data are sampled the first clock when \overline{DLE} is active (low), and should remain stable until \overline{DLE} goes inactive, OR until \overline{RDY} is asserted (low), whichever comes first.

In some configurations, \overline{DLE} may not be provided, and can be permanently tied low. In this case, the independent data timing will occur exactly one clock after the \overline{ADS} signal is first sampled low, OR on the same clock as \overline{BGT} is first sampled low, whichever occurs later. The data bus will be assumed stable at this time. 82489DX recognize that independent \overline{DLE} timing is not provided by sampling a low state of \overline{DLE} at the time \overline{ADS} is first sampled low.

Cycle completion is signaled by the \overline{RDY} signal. Its relative positioning on any of the timing diagrams does NOT imply the number of clock cycles required for an access. \overline{RDY} is delayed as needed in order for the 82489DX to complete the cycle. It is then asserted (low) for one clock cycle and then deasserted. Again, the next \overline{ADS} cannot start until after \overline{RDY} has been driven low. \overline{ADS} must return to an inactive high state before the next cycle can be issued. It is highly recommended not to have, \overline{ADS} more than one clock wide.

2

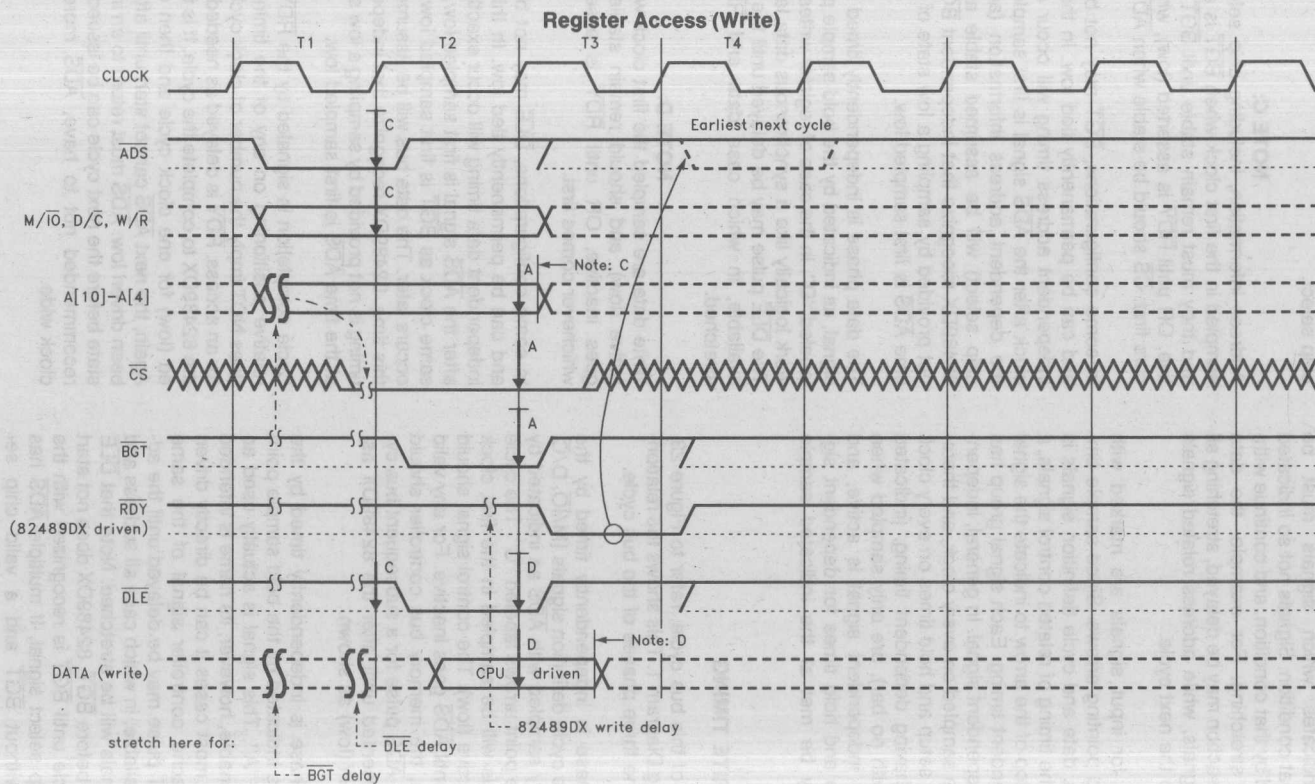
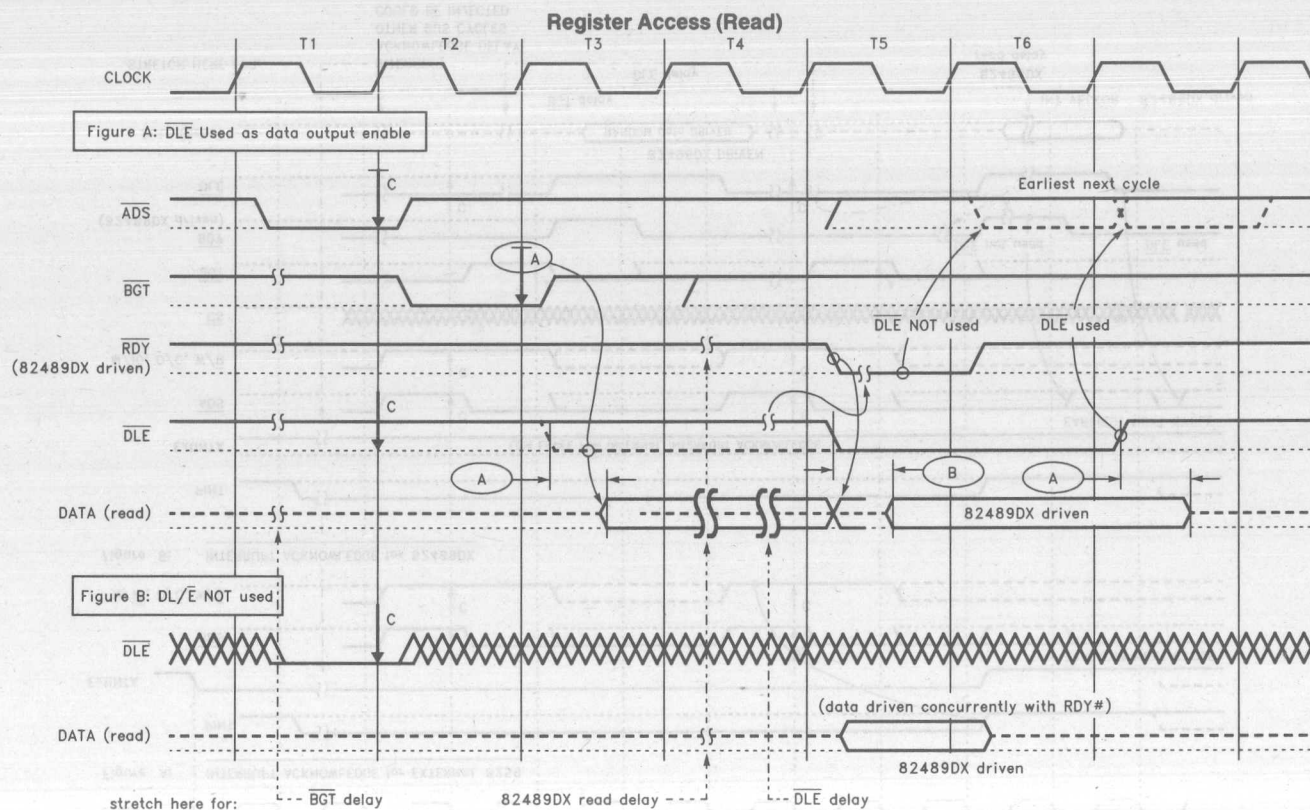


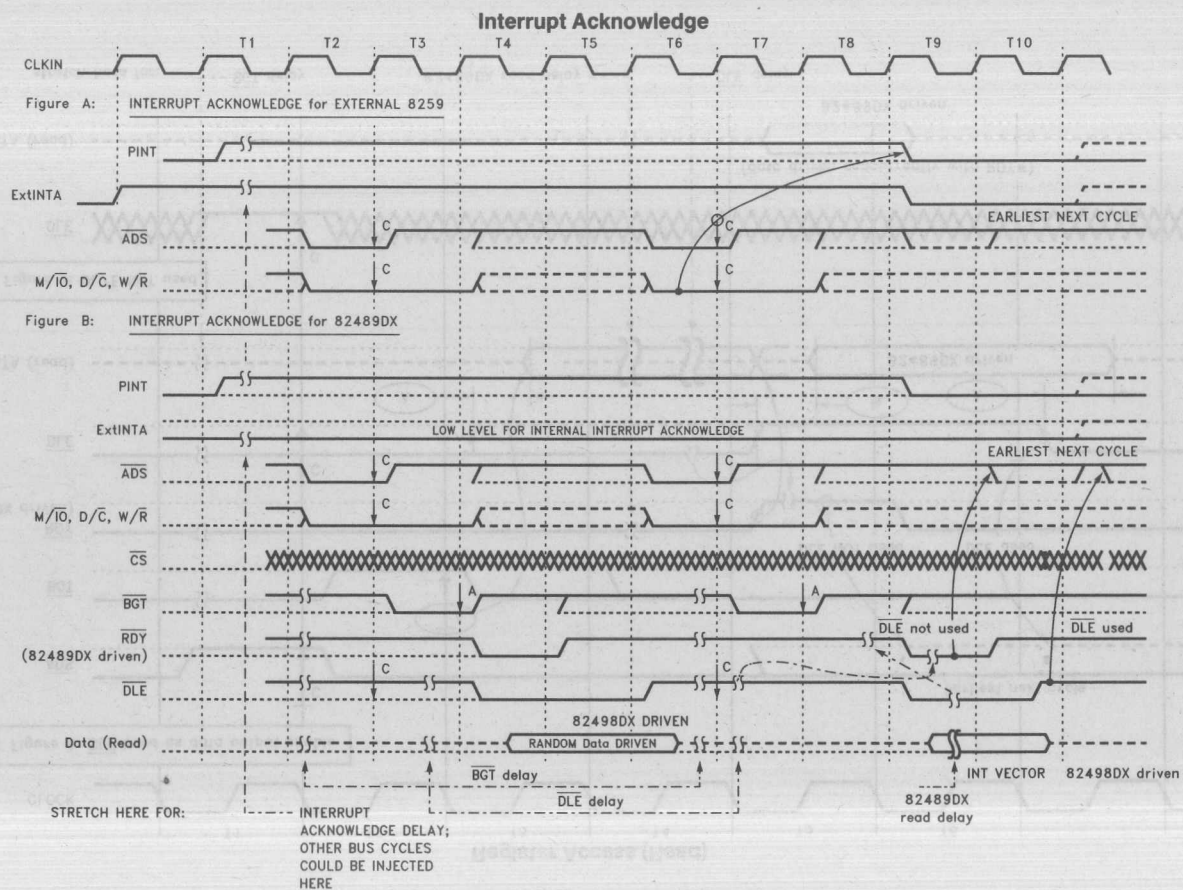
Figure 23. Timing Diagram 1

Figure 24. Timing Diagram 2



290446-14

Figure 25. Timing Diagram 3



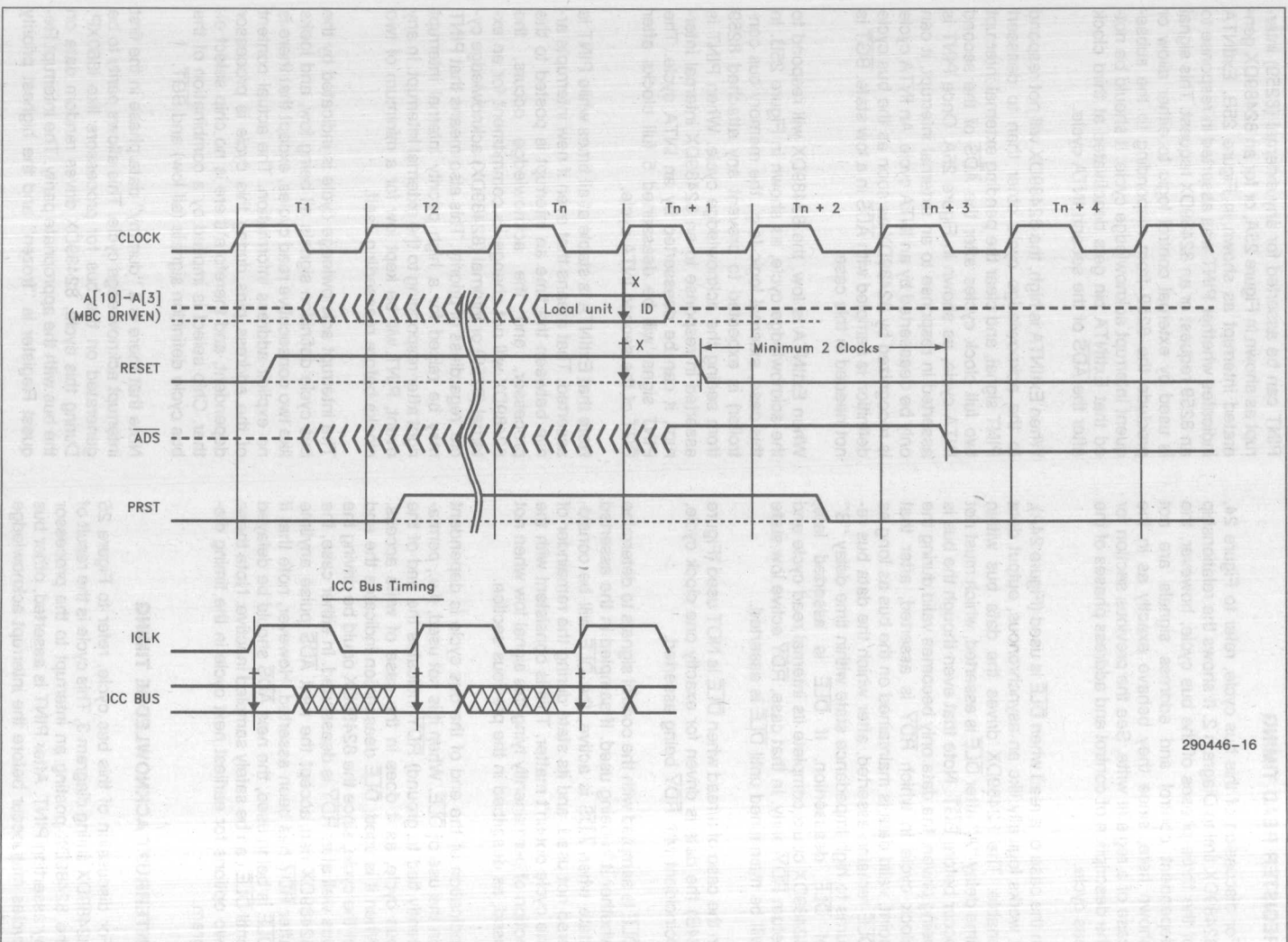


Figure 26. Timing Diagram 4

REGISTER READ TIMING

For discussion of this bus cycle, refer to Figure 24, 82489DX timing Diagram 2. It shows the relationship of the three phases of the bus cycle, however, the dependent control and address signals are not shown here, since they behave exactly as in the case of a register write. See the previous section for the description of control and address phases of the bus cycle.

In the case of a read when \overline{DLE} is used (Figure 24A), it works logically like an asynchronous, output data enable. The 82489DX drives the data bus within time delay "A" after \overline{DLE} is asserted, which must not occur before \overline{BGT} . Note that even though the bus is being driven, the data only becomes valid during the clock cycle in which \overline{RDY} is asserted, after that point, valid data is maintained on the bus as long as \overline{DLE} remains asserted, after which the data bus returns to high impedance state within time delay "B" of \overline{DLE} deassertion. If \overline{DLE} is asserted late, 82489DX could complete its internal read cycle and return \overline{RDY} early. In that case, \overline{RDY} active low state will be maintained until \overline{DLE} is asserted.

In the case of a read when \overline{DLE} is NOT used (Figure 24B) the data is driven for exactly one clock cycle, coincident with \overline{RDY} being asserted.

\overline{DLE} is sampled with the control signals to determine whether it is being used. If sampled in the asserted state when \overline{ADS} is active, the \overline{DLE} will be considered not used, and its state during the remainder of the cycle doesn't matter. This is consistent with the notion of permanently tying this signal low when not used, as described in the previous section.

Indication of the end of the bus cycle is dependent on the use of \overline{DLE} . When it is not used, (i.e., permanently tied to ground) \overline{RDY} indicates the end of the bus cycle, as it does in the case of write access. When it is used, \overline{DLE} deassertion indicates the end of the cycle, since the 82489DX could be driving the bus well after \overline{RDY} is deasserted. In either case, the 82489DX can accept the next \overline{ADS} pulse anytime after \overline{RDY} has been asserted. However, note that if \overline{DLE} is being used, the next \overline{ADS} should be delayed until \overline{DLE} can be safely sampled inactive. Note these two options for earliest next cycle in the timing diagram.

INTERRUPT ACKNOWLEDGE TIMING

For discussion of this bus cycle, refer to Figure 25 82489DX timing diagram 3. This cycle is the result of the 82489DX posting an interrupt to the processor by asserting PINT. After PINT is asserted, other bus cycles may occur before the interrupt acknowledge cycle.

PINT can be asserted for any external (8259) interrupt as shown in Figure 25A, or for an 82489DX generated interrupt as shown in Figure 25B. ExtINTA indicates whether PINT was asserted in response to an 8259 request or an 82489DX request. This signal is used by external control logic to either allow or preclude the 8259 from responding to the subsequent interrupt acknowledge cycle. It should be noted that ExtINTA pin gets deactivated at third clock after the \overline{ADS} of the second INTA cycle.

When ExtINTA is high, the 82489DX will not respond to the acknowledge cycle other than to deassert PINT signal, and clear the pending external interrupt two full clock cycles after the \overline{ADS} of the second INTA cycle, as shown in Figure 25A. Once PINT is asserted in response to an external interrupt, it can only be deasserted by an INTA cycle. An INTA cycle is recognized by 82489DX as soon as the bus cycle definition is sampled with \overline{ADS} in a low state. \overline{BGT} is not needed in this case.

When ExtINTA is low, the 82489DX will respond to the acknowledge cycle, as shown in Figure 25B. In this case, external logic (e.g., the memory bus controller) is expected to prevent any attached 8259 from seeing the acknowledge cycle. When PINT is asserted in response to an 82489DX internal interrupt, it can be deasserted by an INTA cycle. The PINT signal will be deasserted 5 full clocks after \overline{BGT} of the second INTA cycle.

Note that ExtINTA is stable at all times while PINT is asserted. That means that even if new interrupts arrive between the time an interrupt is posted to the processor, and the acknowledge occurs, the 82489DX will not change its commitment for an external (8259) or internal (82489DX) acknowledge cycle, regardless of priority. This also means that PINT may be raised for a high priority internal interrupt right after responding to the external interrupt. In any event, PINT will be kept low for a minimum of two clocks before reasserting itself.

The interrupt acknowledge cycle is indicated by the bus cycle definition signals all being low, and looks like two consecutive read cycles, except that there is no explicit address information. The actual content of the address pins during this cycle is processor dependent, and therefore there is no chip select either. Chip select is implied by a combination of the bus cycle definition signals (all low) and \overline{BGT} .

Note that there is a "dummy" data phase in the first interrupt acknowledge cycle. This allows parity to be generated on the bus for processors like i860XP. During this cycle, 82489DX drives random data on the bus with the appropriate parity. The interrupt Request Register is "frozen" and the highest priority

pending interrupt vector is returned to the processor in the second acknowledge cycle.

The second acknowledge cycle has a complete data phase with timings identical to those of an ordinary register read. The data returned is the vector of the highest priority internally pending 82489DX interrupt, or the spurious interrupt vector, if there is no interrupt pending higher than the current processor priority.

Note that the timing diagram shows \overline{DLE} being used (sampled high during \overline{ADS}). However, just as in a normal read cycle, the option exists not to use \overline{DLE} (i.e., permanently tied to ground).

RESET AND MISCELLANEOUS TIMING

For discussion of this bus cycle, refer to Figure 26 82489DX Timing Diagram 4. It shows the 82489DX reset cycle, the timing of some related signals, and the ICC bus. The RESET input has a setup and hold time to the system clock edge, CLKIN, as do other independently timed signals. The RESET signal will reset the two asynchronous system on the chip, namely the ICC bus unit running synchronously to the ICLK and all the other unit running synchronous to the system clock, CLKIN. RESET must meet the minimum reset time with respect to both clocks and there should be at least one ICLK rising edge during reset. The TAP controller should also be initialized.

During reset, an eight-bit 82489DX Local Unit ID can be optionally initialized. Eight address lines, A10–A3 are sampled on every clock edge while RESET is asserted. The last sample remains in the 82489DX Local Unit ID register after reset. Alternatively, the 82489DX Local Unit ID can be loaded with a register write as part of software initialization, before 82489DX operation is started. In any event, the register must be initialized before the 82489DX can communicate on the ICC bus, including sending/receiving RESET messages. All valid signal to 82489DX should wait at least two full clocks after RESET is deasserted.

The PRST signal (reset output) is asserted both with RESET input, or under software control. Its on-off delay times are relative to the rising clock edge. The duration of PRST under software control is defined by the software itself. Also note that the PNMI pin has the same timing as does PRST when the latter is software controlled.

The ICC bus signals are both input and output on each cycle. Setup, hold and delay times are all measured with respect to the ICC bus Clock ICLK which has no relationship to the Processor clock on which the remainder of the 82489DX runs. This means

that the ICC bus is independently sampled on each ICLK edge, as shown. It also implies that largest possible hold time will not exceed the minimum delay time.

After reset, all 82489DX registers are reset to “0” state. The mask bits in the local vector table and the redirection table are reset to “1” state to mask out all interrupts. All reserved bits are all wired to “0” state permanently on chip.

10.0 BOUNDARY SCAN DESCRIPTION

The 82489DX is equipped with the JTAG boundary scan standard. This feature allows the user to test the interconnections between 82489DX and the external hardware once they have been assembled onto a printed circuit board or other substrate. In addition to the JTAG mandatory instruction set, 82489DX also provides the INTEST instruction which allows static testing of the on-chip logic.

The detailed information related to the IEEE Std 1149.1-1990 (the JTAG standard) can be obtained from the reference document *IEEE Standard Test Access Port and Boundary Scan Architecture (IEEE Std 1149.1-1990)*.

10.1 Boundary Scan Architecture

The boundary scan logic contains the following elements:

- **Five Test Access Ports (TAP):** They are labeled as \overline{trst} , \overline{tck} , \overline{tdi} , \overline{tdo} and \overline{tms} . All ports are input pins except \overline{tdo} , which is a tri-state output pin.
- **A TAP Controller:** The logic is used to control the boundary scan activity.
- **82489DX Device ID Register:** This is a 32-bit read-only register. The DID can be shifted out in ascending order to the \overline{tdo} pin.
- **JTAG Instruction Register (IR):** This is a 4-bit register which accepts instruction code shifted in from the \overline{tdi} pin. The opcode stored in the IR register is used to control operation.
- **Boundary Scan Register:** This is a 137 stages scan path which connects almost all 82489DX signal pins for boundary scan purposes.
- **Bypass Register:** This register simply allows the data which goes into \overline{tdi} pin to be shifted out directly from \overline{tdo} .

The following block diagram illustrates the implementation of the JTAG architecture in the 82489DX design.

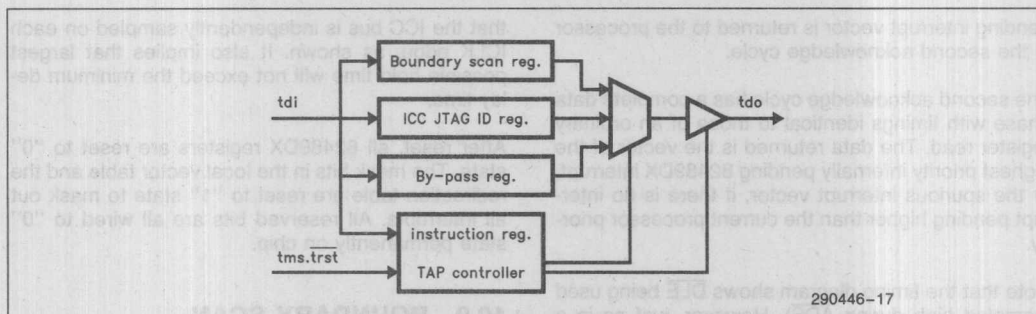


Figure 27. Block Diagram of the JTAG Architecture

Test Access Ports

- trst** TAP controller master reset pin. When $\overline{\text{trst}}$ is low, the TAP controller's state machine will be reset to "test-logic-reset" state asynchronously. This pin is tied to a weak internal pull-up for keeping to be a logical 1 when not driven.
- tck** This is the test logic clock. The test logic will change state on the rising edge of the tck.
- tdi** Test data input. Data is shifted into the tdi pin on the rising edge of tck. This pin is tied to a weak internal pull-up for keeping it to be a logical 1 when not driven.
- tms** Test mode select. This pin is used to select the state of the TAP controller. This pin is synchronous to the rising edge of the tck. This pin is tied to a weak internal pull-up for keeping it to be a logical 1 when not driven.

tdo Test data output. This is a tri-state pin which allows the data to be shifted out.

TAP CONTROLLER

The TAP controller in 82489DX is implemented to conform the IEEE1149.1 standard. The TAP controller is a single phase clock, synchronous finite state machine. It controls the sequence of the operation of the test logic.

The value of the test mode state (tms) pin at a rising edge of tck controls the sequence of the state changes. The state diagram for the TAP controller is shown in Figure 28. Test designers must consider the operation of the state machine in order to have the correct sequence of value to drive on tms.

The behavior of the TAP controller and other test logic in each of the controller states is briefly described as follows.

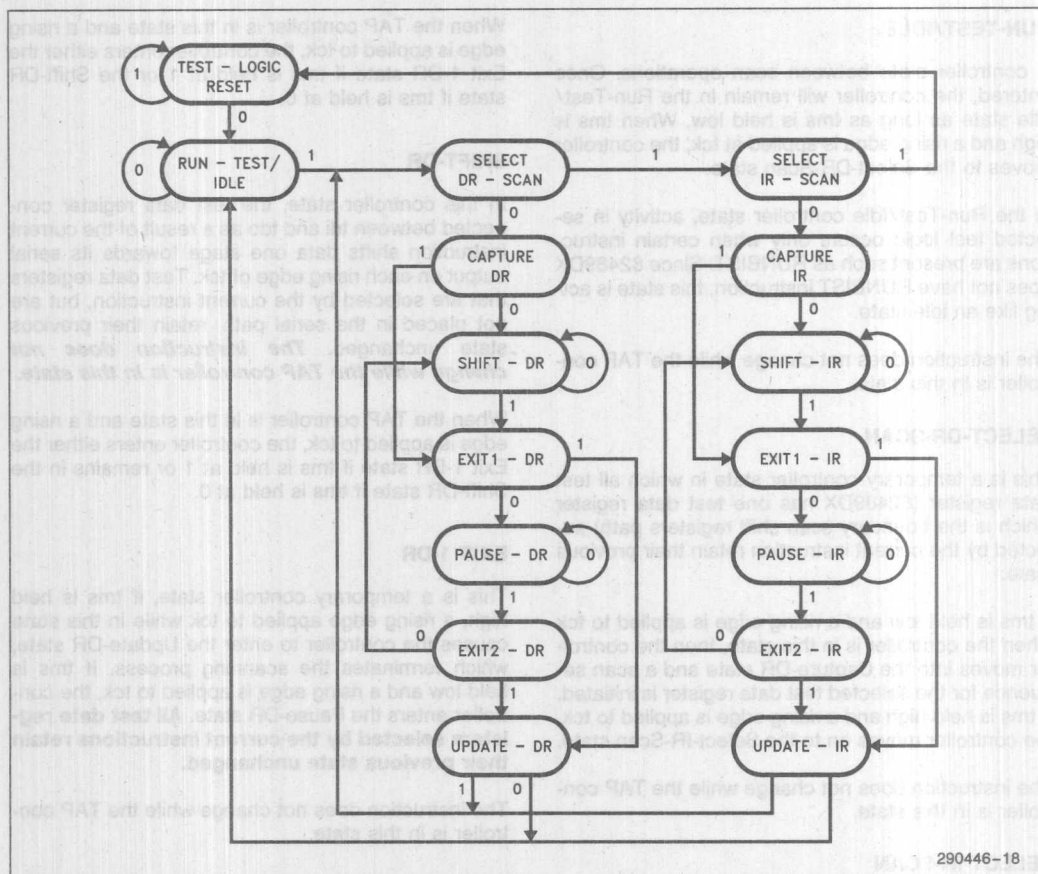


Figure 28. TAP Controller State Diagram

TEST-LOGIC-RESET

The test logic is disabled so that normal operation of the on-chip system logic (i.e., in response to stimuli received through the system pins only) can continue unhindered. This is achieved by initializing the instruction register to contain the IDCODE instruction. No matter what the original state of the controller, it will enter Test-Logic-Reset when tms is held high for at least five rising edges of tck. The controller remains in this state while tms is high.

If the controller should leave the Test-Logic-Reset controller state as a result of an erroneous low signal on the tms line at the time of the rising edge on tck (for example, a glitch due to external interfer-

ence), it will return to the Test-Logic-Reset state following three rising edges of tck with the tms line at the intended high logic level. The operation of the test logic is such that no disturbance is caused to on-chip system logic operation as the results of such an error. On leaving the Test-Logic-Reset controller state, the controller moves into the Run-Test/Idle controller state where no action will occur because the current instruction has been set to select operation of the device identification register. The test logic is also inactive in the Select-DR-Scan and Select-IR-Scan controller states.

Note that the TAP controller will also be forced to the Test-Logic-Reset controller state asynchronously by applying a low logic level at trst.

RUN-TEST/IDLE

A controller state between scan operations. Once entered, the controller will remain in the Run-Test/Idle state as long as tms is held low. When tms is high and a rising edge is applied at tck, the controller moves to the Select-DR-Scan state.

In the Run-Test/Idle controller state, activity in selected test logic occurs only when certain instructions are present such as RUNBIST. Since 82489DX does not have RUNBIST instruction, this state is acting like an idle state.

The instruction does not change while the TAP controller is in this state.

SELECT-DR-SCAN

This is a temporary controller state in which all test data register (82489DX has one test data register which is the boundary scan shift registers path) selected by the current instruction retain their previous state.

If tms is held low and a rising edge is applied to tck when the controller is in this state, then the controller moves into the Capture-DR state and a scan sequence for the selected test data register is initiated. If tms is held high and a rising edge is applied to tck, the controller moves on to the Select-IR-Scan state.

The instruction does not change while the TAP controller is in this state.

SELECT-IR-SCAN

This is a temporary controller state in which all test data registers selected by the current instructing retain their previous state.

If tms is held low and a rising edge is applied to tck when the controller is in this state, then the controller moves into the Capture-IR state and a scan sequence for the instruction register is initiated. If tms is held high and a rising edge is applied to tck, the controller returns to the Test-Logic-Reset state. **The instruction does not change while the TAP controller is in this state**

CAPTURE-DR

In this controller state data may be parallel-loaded into test data registers selected by the current instruction on the rising edge of tck. If a test data register selected by the current instruction does not have a parallel input, or if capturing is not required for the selected test, then the register retains its previous state unchanged. **The instruction does not change while the TAP controller is in this state.**

When the TAP controller is in this state and a rising edge is applied to tck, the controller enters either the Exit 1-DR state if tms is held at 1 or the Shift-DR state if tms is held at 0.

SHIFT-DR

In this controller state, the test data register connected between tdi and tdo as a result of the current instruction shifts data one stage towards its serial output on each rising edge of tck. Test data registers that are selected by the current instruction, but are not placed in the serial path, retain their previous state unchanged. **The instruction does not change while the TAP controller is in this state.**

When the TAP controller is in this state and a rising edge is applied to tck, the controller enters either the Exit 1-DR state if tms is held at 1 or remains in the Shift-DR state if tms is held at 0.

EXIT 1-DR

This is a temporary controller state, if tms is held high, a rising edge applied to tck while in this state causes the controller to enter the Update-DR state, which terminates the scanning process. If tms is held low and a rising edge is applied to tck, the controller enters the Pause-DR state. **All test data registers selected by the current instructions retain their previous state unchanged.**

The instruction does not change while the TAP controller is in this state.

PAUSE-DR

This controller state allows shifting of the test data register in the serial path between tdi and tdo to be temporarily halted. All test data registers selected by the current instruction retain their previous state unchanged.

The controller remains in this state while tms is low. When tms goes high and a rising edge is applied to tck, the controller moves on to the Exit 2-DR state. The instruction does not change while the TAP controller is in this state.

EXIT 2-DR

This is a temporary controller state. If tms is held high and a rising edge is applied to tck while in this state, the scanning process terminates and the TAP controller enters the Update-DR controller state. If tms is held low and a rising edge is applied to tck, the controller enters the Shift-DR state.

All test data registers selected by the current instruction retain their previous state unchanged. The instruction does not change while the TAP controller is in this state.

UPDATE-DR

Some test data registers may be provided with a latched parallel output to prevent changes at the parallel output while data is shifted in the associated shift-register path in response to certain instructions (e.g., EXTENT, INTEST, and RUNBIST). Data is latched onto the parallel output of these test data registers from the shift-register path on the falling edge of tck in the Update-DR controller state. The data held at the latched parallel output should not change other than in this controller state unless operation during the execution of a self test is required (e.g., during the Run-Test/Idle controller state in response to a design-specific public instruction).

All shift-register stages in test data registers selected by the current instruction retain their previous state unchanged. **The instruction does not change while the TAP controller is in this state.**

When the TAP controller is in this state and a rising edge is applied to tck, the controller enters either the Select-DR-Scan state if tms is held at 1 or the Run-Test/Idle state if tms is held at 0.

CAPTURE-IR

In this controller state the shift-register contained in the instruction register loads a pattern of fixed logic values on the rising edge of tck. In addition, design-specific data may be loaded into shift-register stages that are not required to be set to fixed values.

Test data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state.

When the TAP controller is in this state and rising edge is applied to tck, the controller enters either the Exit 1-IR state tms is held at 1 or the Shift-IR state if tms is held at 0.

SHIFT-IR

In this controller state the shift-register contained in the instruction register is connected between tdi and tdo and shifts data one stage towards its serial output on each rising edge of tck.

Test data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state. When the TAP controller is in this state and a rising edge is applied to tck, the controller enters either the Exit1-IR state if tms is held at 1 or remains in Shift-IR state if tms is held at 0.

EXIT 1-IR

This is a temporary controller state. If tms is held high, a rising edge applied to tck while in this state causes the controller to enter the Update-IR state, which terminates the scanning process. If tms is held low and a rising edge is applied to tck, the controller enters the Pause-IR state.

Test data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state and the instruction register retains its state.

PAUSE-IR

This controller state allows shifting of the instruction register to be halted temporarily.

Test data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state and the instruction register retains its state.

The controller remains in this state while tms is low. When tms goes high and a rising edge is applied to tck, the controller moves on to the Exit 2-IR state.

EXIT 2-IR

This is a temporary controller state. If tms is held high and a rising edge is applied to tck while in this state, termination of the scanning process results, and the TAP controller enters the Update-IR controller state. If tms is held low and a rising edge is applied to tck, the controller enters the Shift-IR state.

Test data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state and the instruction register retains its state.

UPDATE-IR

The instruction shifted into the instruction register is latched onto the parallel output from the shift-register path on the falling edge of tck in this controller state. Once the new instruction has been latched, it

becomes the current instruction. **Test data registers selected by the current instruction retain their previous state.**

When the TAP controller is in this state and a rising edge is applied to tck, the controller enters the Select-DR-Scan states if tms is held at 1 or the Run-Test/Idle state if tms is held at 0.

INSTRUCTION REGISTER

The function of the instruction register is to select the operating mode of the test logic. For instance, read the ID register, or capture the 82489DX output signals. 82489DX has implemented 4 instructions.

Instruction	Mandatory/Optional	Opcode
bypass	m	1 1 1 1
extest	m	0 0 0 0
sample/preload	m	0 0 0 1
idcode	m	0 0 1 0
reserved	o	1 0 0 1

Bypass Instruction

The bypass instruction selects the bypass register to be connected to tdi and tdo, effectively bypassing the test logic on the 82489DX boundary scan path and reducing the shift length to be on one bit. Note that an open circuit fault in the board level test data path will cause the bypass register to be selected following an instruction scan cycle due to the internal pull-up on the tdi pin. This has been done to prevent any unwanted interference with the proper operation of the system logic.

Extest Instruction

The extest instruction allows testing of circuitry external to the component package, typically board interconnects. It does so by driving the values loaded into the 82489DX's boundary scan register out on the output pins corresponding to each boundary scan cell and capturing the values on 82489DX's input pins to be loaded into their corresponding boundary scan register locations. I/O pins are selected as input or output depending on the value located into the output control cell. Values shifted into input latch in the boundary scan register are never used by the internal logic of the 82489DX.

NOTE:

82489DX must be reset after extest instruction has been executed.

Sample/Preload Instruction

The sample/preload instruction has two functions that it can perform. When the TAP controller is in the CAPTURE-DR state, the sample/preload instruction allows a snap-shot of the normal operation of the 82489DX without interfering with that normal operation. The instruction causes boundary scan register cells associated with outputs to sample the value being driven into the 82489DX. On both outputs and inputs the sampling occurs on the rising edge of tck. When preloads data into the 82489DX pins to be driven to the board by executing the extest instruction. Data is preloaded to the pins from the boundary scan register on the falling edge of tck.

idcode Instruction

The idcode instruction selects the device identification register to be connected to tdi and tdo, allowing the device ID code to be shifted out of the device on tdo. Note that the bit stream shifted into tdi will appear on tdo after all 32 bits of the DID has been shifted out.

DEVICE IDENTIFICATION REGISTER (DID)

The device identification is a 32 bits number which can be read by the external hardware by using the idcode instruction. The 82489DX device ID is assigned to 1489A013 (hex). This is subject to change. The upper 4 bits of DID may be changed for different version. The 16-bit number (bit 27-bit 12) 489A (hex) is the part ID. The lower 12 bits are the manufacturer ID for Intel which must be 013 (hex).

BOUNDARY SCAN REGISTER

82489DX has only one test data register, i.e., the boundary scan register. The boundary scan register is a single shift register path containing the boundary scan cells that are connected to all signal input and output pins of the 82489DX. There are three generic type of boundary scan cells—input, output, and bi-directional. For each input only cell, one stage of shift register is added to the boundary scan path.

All output pins will become tri-stateable when boundary scan is activated, regardless whether they are tri-stateable or not in the normal operation. To explain further, the user will enable/disable an out-

put driver with a specific tri-state control cell in the scan path. The user must shift in a proper control signal for these tri-state control cells in the scan path.

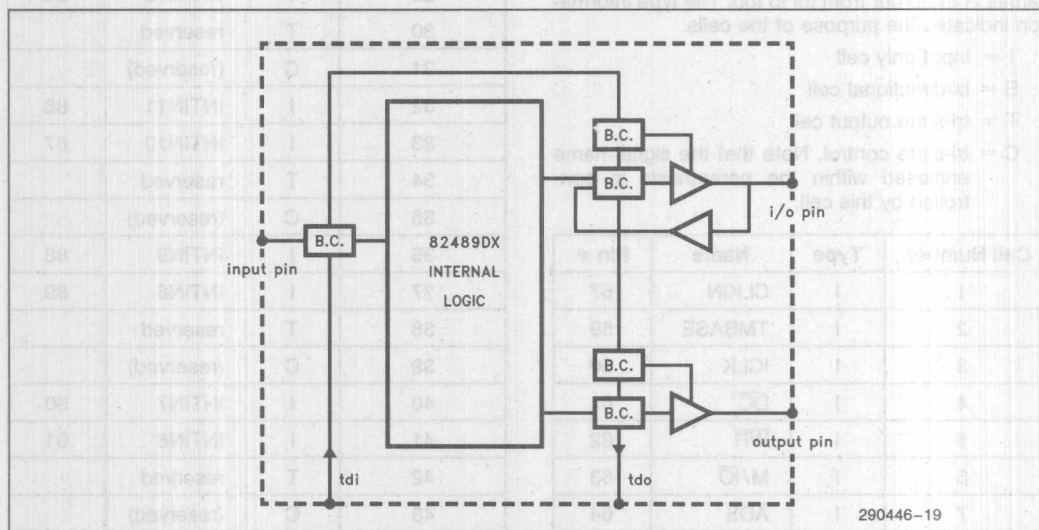


Figure 29. Logical Structure of Boundary Scan Register

2

FROM tdi TO tdo

The following table is a list of the boundary scan cell names in the order from tdi to tdo. The type information indicates the purpose of the cells.

I = input only cell

B = bi-directional cell

T = tri-state output cell

C = tri-state control. Note that the signal name enclosed within the parenthesis is controlled by this cell.

Cell Number	Type	Name	Pin #
1	I	CLKIN	57
2	I	TMBASE	59
3	I	ICLK	60
4	I	DC	61
5	I	WR	62
6	I	M/ \overline{IO}	63
7	I	ADS	64
8	I	RESET	65
9	I	BGT	66
10	I	reserved	70
11	I	reserved	71
12	I	reserved	72
13	I	\overline{DLE}	73
14	I	\overline{CS}	74
15	I	reserved	75
16	I	MBI3	76
17	I	MBI2	77
18	I	MBI1	78
19	I	MBI0	79
20	I	LINTIN1	80
21	I	LINTIN0	81
22	T	reserved	
23	C	(reserved)	
24	I	INTIN15	82
25	I	INTIN14	83
26	T	reserved	
27	C	(reserved)	

Cell Number	Type	Name	Pin #
28	I	INTIN13	84
29	I	INTIN12	85
30	T	reserved	
31	C	(reserved)	
32	I	INTIN11	86
33	I	INTIN10	87
34	T	reserved	
35	C	(reserved)	
36	I	INTIN9	88
37	I	INTIN8	89
38	T	reserved	
39	C	(reserved)	
40	I	INTIN7	90
41	I	INTIN6	91
42	T	reserved	
43	C	(reserved)	
44	I	INTIN5	92
45	I	INTIN4	93
46	T	reserved	
47	C	(reserved)	
48	I	INTIN3	94
49	I	INTIN2	95
50	T	reserved	
51	C	(reserved)	
52	I	INTIN1	96
53	I	INTIN0	97
54	I	reserved	
55	B	DP3	101
56	C	(DP[3:0])	
57	B	DP2	102
58	B	DP1	103
59	B	DP0	104
60	T	reserved	
61	C	(reserved)	
62	B	D31	105
63	C	(D[31:0])	
64	B	D30	107

Cell Number	Type	Name	Pin #
65	B	D29	109
66	B	D28	110
67	T	reserved	
66	C	(reserved)	
69	B	D27	111
70	B	D26	112
71	T	reserved	
72	C	(reserved)	
73	B	D25	114
74	B	D24	115
75	B	D23	116
76	T	reserved	
77	C	(reserved)	
78	B	D22	118
79	B	D21	119
80	B	D20	121
81	B	D19	122
82	B	D18	123
83	B	D17	124
84	B	D16	125
85	B	D15	128
86	B	D14	129
87	B	D13	130
88	B	D12	131
89	B	D11	2
90	T	reserved	
91	C	(reserved)	
92	B	D10	3
93	B	D9	4
94	T	reserved	
95	C	(reserved)	
96	B	D8	7
97	B	D7	8
98	B	D6	9
99	B	D5	11
100	B	D4	12
101	B	D3	13

Cell Number	Type	Name	Pin #
102	B	D2	14
103	B	D1	16
104	B	D0	18
105	I	reserved	19
106	B	reserved	20
107	C	(cell106, A[10:3])	
108	B	A10	21
109	B	A9	22
110	B	A8	24
111	B	A7	26
112	B	A6	27
113	B	A5	28
114	B	A4	29
115	B	A3	31
116	T	reserved	34
117	C	reserved	
118	T	PINT	35
119	C	(PINT)	
120	T	PNMI	37
121	C	(PNMI)	
122	T	PRST	38
123	C	(PRST)	
124	T	ExtINTA	41
125	C	(reserved)	
126	T	reserved	42
127	C	(reserved)	
128	T	$\overline{\text{RDY}}$	43
129	C	($\overline{\text{RDY}}$)	
130	T	MBO3	45
131	C	(MBO3)	
132	T	MBO2	48
133	C	(MBO2)	
134	T	MBO1	49
135	C	(MBO1)	
136	T	MBO0	51
137	C	(MBO0)	

BYPASS REGISTER

The bypass register is simply a 1-bit shift register which connects between the tdi and tdo. When selected by using the bypass instruction, the data shifted into tdi will be shifted out from tdo one tck clock later.

JTAG TAP Controller Initialization

The TAP controller must be reset to test-logic-reset state when 82489DX is first powered up. There are two ways to reset the TAP controller:

1. Assert $\overline{\text{trst}}$ to be 0, it will reset the TAP controller asynchronously.

2. Assert tms to be 1, and clock the TAP controller at least five times, the TAP controller will be reset after the fifth rising edge of the tck.

After reset, the idcode instruction is loaded into the IR automatically.

Note that the tms and $\overline{\text{trst}}$ pins both have an internal weak pull-up device to keep them to be logic 1 level. Therefore the user can simply apply 5 clocks at the tck input to reset the TAP controller. If the TAP controller is not reset properly, 82489DX may not function because the boundary scan logic might be active which will impact the signals flow in and out to the chip.

32	AT	B	117
33	AB	B	118
34	AS	B	119
35	AT	B	120
36	AS	B	121
37	reserved	T	122
38	reserved	C	123
39	PRT	T	124
40	(PRT)	C	125
41	PRT	T	126
42	(PRT)	C	127
43	PRST	T	128
44	(PRST)	C	129
45	PRST	T	130
46	(PRST)	C	131
47	PRST	T	132
48	(PRST)	C	133
49	PRST	T	134
50	(PRST)	C	135
51	PRST	T	136
52	(PRST)	C	137

53	DS0	B	138
54	DS1	B	139
55	DS2	B	140
56	reserved	T	141
57	(reserved)	C	142
58	DS3	B	143
59	DS4	B	144
60	DS5	B	145
61	DS6	B	146
62	DS7	B	147
63	DS8	B	148
64	DS9	B	149
65	DS10	B	150
66	DS11	B	151
67	DS12	B	152
68	DS13	B	153
69	DS14	B	154
70	DS15	B	155
71	DS16	B	156
72	DS17	B	157
73	DS18	B	158
74	DS19	B	159
75	DS20	B	160
76	DS21	B	161
77	DS22	B	162
78	DS23	B	163
79	DS24	B	164
80	DS25	B	165
81	DS26	B	166
82	DS27	B	167
83	DS28	B	168
84	DS29	B	169
85	DS30	B	170
86	DS31	B	171
87	DS32	B	172
88	DS33	B	173
89	DS34	B	174
90	DS35	B	175
91	DS36	B	176
92	DS37	B	177
93	DS38	B	178
94	DS39	B	179
95	DS40	B	180
96	DS41	B	181
97	DS42	B	182
98	DS43	B	183
99	DS44	B	184
100	DS45	B	185
101	DS46	B	186

11.0 ELECTRICAL CHARACTERISTICS

11.1 D.C. Specifications

ABSOLUTE MAXIMUM RATINGS

Case Temperature Under Bias . . . -65°C to $+110^{\circ}\text{C}$
Storage Temperature -65°C to $+150^{\circ}\text{C}$
Voltage on Any Pin
with Respect to Ground -0.5 to $V_{\text{CC}} + 0.5$

NOTICE: This data sheet contains information on products in the sampling and initial production phases of development. The specifications are subject to change without notice. Verify with your local Intel Sales office that you have the latest data sheet before finalizing a design.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

$V_{\text{CC}} = 5\text{V} \pm 5\%$; $T_{\text{C}} = 0^{\circ}\text{C}$ to $+85^{\circ}\text{C}$

Symbol	Parameter	Min (ns)	Max	Units	Notes
V_{IL}	Input LOW Voltage (TTL)	-0.3	$+0.8$	V	
V_{IH}	Input HIGH Voltage (TTL)	2.0	$V_{\text{CC}} + 0.3$	V	
V_{OL}	Output LOW Voltage (TTL)		$+0.45$	V	(Note 1)
V_{QH}	Output HIGH Voltage (TTL)	2.4		V	(Note 2)
I_{CC}	33 MHz Power Supply Current		200	mA	
I_{LI}	Input Leakage Current		15	μA	
I_{LL}	Input Leakage Current		-600	μA	(Note 5)
I_{LH}	Output Leakage Current		600	μA	(Note 4)
I_{LO}	Output Leakage Current		15	μA	(Note 3)
C_{IN}	Input Capacitance		3	pF	
C_{O}	I/O or Output Capacitance		6	pF	
C_{CLKIN}	Clock Capacitance		3	pF	
I_{MLO}	ICC Bus Output Low Current		4	mA	(Note 6)
C_{MC}	ICC Bus Total Capacitance		100	pF	
V_{MH}	ICC Bus Input High (TTL)	2.0	$V_{\text{CC}} + 0.3$	V	
V_{ML}	ICC Bus Input Low (TTL)	-0.3	$+0.8$	V	

NOTES:

1. This parameter is measured with current load of 4 mA.
2. This parameter is measured with current load of 1.0 mA.
3. This parameter is for output without pulldown.
4. This parameter is for tri-state output with pulldown and $V_{\text{OH}} = 3.0\text{V}$.
5. This parameter is for input with pullup at $V_{\text{IL}} = 0\text{V}$.
6. ICC bus output low current is measured at 0.6V.

11.2 A.C. Specifications

A.C. Parameters Referencing 33 MHz System Clock

$V_{CC} = 5V \pm 5\%$; $T_C = 0^\circ C$ to $+85^\circ C$

Symbol	Parameter	Ref. Fig.	Load (pF)	Min (ns)	Max (ns)	Notes
t _c	CLKIN Period	30		30	100	(Note 1)
t ₁	CLKIN High Time	30		5		
t ₂	CLKIN Low Time	30		5		
t ₃	CLKIN Rise Time	30			3	(Note 2)
t ₄	CLKIN Fall Time	30			3	(Note 2)
t ₅	\overline{ADS} , BGT, \overline{DLE} , M/ \overline{IO} , D/ \overline{C} , W/ \overline{R} , \overline{CS} Setup Time	31		8		
t ₆	D31–D0, DP3–DP0, A9–A3 Setup Time	31		8		
t ₈	\overline{ADS} , BGT, \overline{DLE} , M/ \overline{IO} , D/ \overline{C} , W/ \overline{R} , \overline{CS} Hold Time	31		5		
t ₁₀	D31–D0, DP3–DP0, A9–A3 Hold Time	31		5		
t ₁₁	D31–D0, DP3–DP0, Valid Delay	30	50		18	
t ₁₂	D31–D0, DP3–DP0, Low-Z Delay When \overline{DLE} is Not Used	32	50	3		(Note 7)
t ₁₃	D31–D0, DP3–DP0, High-Z Delay When \overline{DLE} is Not Used	32	50		14	(Note 7)
t ₁₄	D31–D0, DP3–DP0 Enable Delay When \overline{DLE} is Used	33	50	3	42	
t ₁₅	D31–D0, DP3–DP0 Disable Delay When \overline{DLE} is Used	33	50	3	14	
t ₂₀	\overline{RDY} Valid Delay	30	50	3	18	
t ₂₁	PRST, PNMI, PINT Valid Delay	30	50	3	34	
t ₂₂	RESET Setup Time	31		8		(Note 5)
t ₂₃	RESET Hold Time	31		5		(Note 5)
	RESET Cycle Time			5 t _c		(Note 3)
				1 t _{ic}		(Note 3)
t ₂₄	INTIN[15:0], LINTIN[1:0] Low Time			10		(Note 6)

All parameters are given in nanoseconds.

TTL Level timing is measured at 1.5V for both "0" and "1" levels.

NOTES:

1. ICC bus clock ICLK period must be at least 5 ns longer than system clock CLKIN for proper synchronization of the internal asynchronous signals.
2. System clock CLKIN measured from 0.8V–2.0V.
3. Minimum Reset cycle is the greater of the two cycle times.
4. Minimum pulse width must be met for valid level to be attained on the DATA or ADDRESS output.
5. Set up and hold time is required for RESET to start at the next rising edge of the clock.
6. INTIN and LINTIN low time is measured from 1.5V of the falling edge to 1.5V of rising edge.
7. Not 100% tested. Guaranteed by design characterization.

Time Base A.C. Parameters

 $V_{CC} = 5V \pm 5\%$; $T_C = 0^{\circ}C$ to $+85^{\circ}C$

Symbol	Parameter	Ref. Fig.	Min (ns)	Max (ns)	Note
tmc	TMBASE Period	35	40	10000	
t30	TMBASE High Time	35	10		
t31	TMBASE Low Time	35	10		
t32	TMBASE Rise Time	35		8	
t33	TMBASE Fall Time	35		8	

TAP Controller A.C. Parameters $V_{CC} = 5V \pm 5\%$; $T_C = 0^{\circ}C$ to $+85^{\circ}C$

Symbol	Parameter	Ref. Fig.	Min (ns)	Max (ns)	Note
ttc	TCK Period	35	40	1000	
t50	TCK High Time	35	10		
t51	TCK Low Time	35	10		
t52	TCK Rise Time	35		8	
t53	TCK Fall Time	35		8	
t54	TDI, TMS, \overline{TRST} Setup Time	34	10		
t55	TDI, TMS, \overline{TRST} Hold Time	34	5		
t56	TDO VALID Delay	34	5	24	(Note 1)
t57	Output Delay in EXTest in EXTEST Mode	34	5	27	(Note 1)
t58	\overline{TRST} Minimum Low Time		10		(Note 2)

All parameters are given in nanoseconds.

TTL level timing is measured at 1.5V for both "0" and "1" levels.

NOTES:

- These parameters are specified for 50 pF load.
- This parameter is measured at 1.5V between the rising and falling edges.

A.C. Parameters for ICC Bus $V_{CC} = 5V \pm 5\%$; $T_C = 0^\circ C$ to $+85^\circ C$

Symbol	Parameter	Ref. Fig.	Min (ns)	Max (ns)	Notes
tic	ICLK Period	35	60		(Note 1)
t40	ICLK High Time	35	20		
t41	ICLK Low Time	35	20		
t42	ICLK Rise Time	35		10	
t43	ICLK Fall Time	35		10	
t44	MBI3-MB10 Setup Time	36	8		
t45	MBI3-MB10 Hold Time	36	5		
t46	MB03-MBO0 VALID Low Delay	36		50	(Note 2)
t47	MB03-MBO0 VALID High-Z Delay	36	5	15	(Note 3)
t48	MB03-MBO0 VALID Low-Z Delay	36	12	25	(Note 3)

All parameters are given in nanoseconds.

TTL level timing is measured at 1.5V for both "0" and "1" levels.

NOTES:

1. MBI3-0 and MBO3-0 timing is tested at 150 ns cycle time.
2. This parameter is specified for 50 pF load.
3. Not 100% tested. Guaranteed by design characterization.

12.0 REGISTER SUMMARY

82489DX registers can be located at any 1 Kbyte boundary in either memory or I/O space for as far as the 82489DX architecture itself is concerned. From a platform standard point of view, it is recommended to locate all 82489DX Local Units in memory space at address 0xFEE0—0000. It is further recommended that all 82489DX I/O Units also be located in memory space; I/O Unit 1 at address 0xFEC0—0000, I/O Unit 2 (if present) at address 0xFEC0—1000, and so on. Chip select for the 82489DX should be based on a full decode of address pins A31-A10.

All directly accessible 82489DX registers are 32 bits wide and are aligned at 128-bit boundaries. The register being accessed is determined by bits 4 through 9 of the address. This is listed in the tables below.

Addresses not listed are reserved by the architecture. The tables also show whether the register is readable and/or writable by software, and what the side effects are of software accessing the register.

After reset, all registers are initialized to all zeroes with the following exceptions, The Local Unit ID field is initialized with data present on the 8 LSB address pins. The Mask bit is initialized to 1 ("masked" state) in all entries in both the local vector table and the redirection table.

For the I/O Unit, only the I/O register select and I/O window registers are directly accessible in the address space. The other I/O unit registers are accessed indirectly through the select and window register.

I/O Unit Registers

Register	Address (9:4)	SW	Side Effects
I/O Register Select	00 0000	W	
I/O Window Register	00 0001		

Register	I/O Reg Select (7:0)	SW	Side Effects
I/O Unit ID Register	0000 0000	rw	
Version Register	0000 0001	r	
Redirection Table [0] (31:0)	0001 0000	rw	
Redirection Table [0] (63:32)	0001 0001	rw	
Redirection Table [1] (31:0)	0001 0010	rw	
Redirection Table [1] (63:32)	0001 0011	rw	
Redirection Table [2] (31:0)	0001 0100	rw	
Redirection Table [2] (63:32)	0001 0101	rw	
Redirection Table [3] (31:0)	0001 0110	rw	
Redirection Table [3] (63:32)	0001 0111	rw	
Redirection Table [4] (31:0)	0001 1000	rw	
Redirection Table [4] (63:32)	0001 1001	rw	
Redirection Table [5] (31:0)	0001 1010	rw	
Redirection Table [5] (63:32)	0001 1011	rw	
Redirection Table [6] (31:0)	0001 1100	rw	
Redirection Table [6] (63:32)	0001 1101	rw	
Redirection Table [7] (31:0)	0001 1110	rw	
Redirection Table [7] (63:32)	0001 1111	rw	
Redirection Table [8] (31:0)	0010 0000	rw	
Redirection Table [8] (63:32)	0010 0001	rw	
Redirection Table [9] (31:0)	0010 0010	rw	
Redirection Table [9] (63:32)	0010 0011	rw	
Redirection Table [10] (31:0)	0010 0100	rw	
Redirection Table [10] (63:32)	0010 0101	rw	
Redirection Table [11] (31:0)	0010 0110	rw	
Redirection Table [11] (63:32)	0010 0111	rw	
Redirection Table [12] (31:0)	0010 1000	rw	
Redirection Table [12] (63:32)	0010 1001	rw	
Redirection Table [13] (31:0)	0010 1010	rw	
Redirection Table [13] (63:32)	0010 1011	rw	
Redirection Table [14] (31:0)	0010 1100	rw	
Redirection Table [14] (63:32)	0010 1101	rw	
Redirection Table [15] (31:0)	0010 1110	rw	
Redirection Table [15] (63:32)	0010 1111	rw	

Registers	Address (9:4)	SW	Side Effects
Local Unit ID Register	00 0010	rw	
Version Register	00 0011	r	
Reserved	00 0100		
Reserved	00 0101		
Reserved	00 0110		
Reserved	00 0111		
Task Priority Register	00 1000	rw	mask intr dispense
Reserved	00 1001		
Reserved	00 1010		
EOI Register	00 1011	rw	prioritization cycle
Remote Register	00 1100	r	
Logical Destination Reg.	00 1101	rw	
Destination Format Reg.	00 1110	rw	
Spurious Vector Register	00 1111	rw	
ISR (31:0)	01 0000	r	
ISR (63:32)	01 0001	r	
ISR (95:64)	01 0010	r	
ISR (127:96)	01 0011	r	
ISR (159:128)	01 0100	r	
ISR (191:160)	01 0101	r	
ISR (223:192)	01 0110	r	
ISR (255:224)	01 0111	r	
TMR (31:0)	01 1000	r	
TMR (63:32)	01 1001	r	
TMR (95:64)	01 1010	r	
TMR (127:96)	01 1011	r	
TMR (159:128)	01 1100	r	
TMR (191:160)	01 1101	r	
TMR (223:192)	01 1110	r	
TMR (255:224)	01 1111	r	
IRR (31:0)	10 0000	r	
IRR (63:32)	10 0001	r	
IRR (95:64)	10 0010	r	
IRR (127:96)	10 0011	r	
IRR (159:128)	10 0100	r	
IRR (191:160)	10 0101	r	

LOCAL UNIT REGISTERS (Continued)

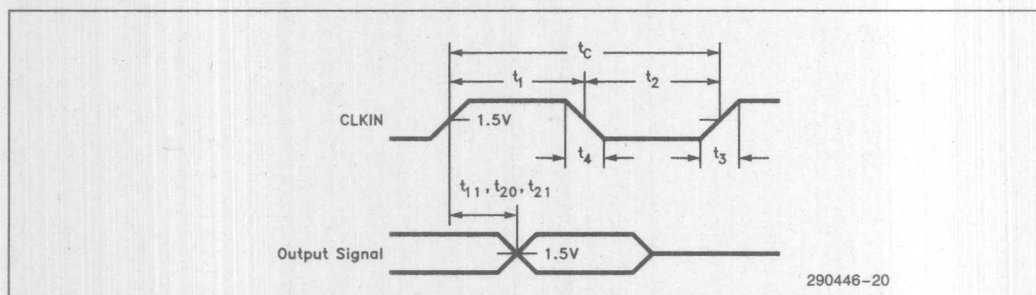
Registers	Address (9:4)	SW	Side Effects
IRR (223:192)	10 0110	r	
IRR (255:224)	10 0111	r	
Intrpt Comnd Reg. (31:0)	11 0000	rw	send interrupt
Intrpt Comnd Reg. (63:32)	11 0001	rw	
Local Vector Table [timer]	11 0010	rw	
Reserved	11 0011		
Reserved	11 0100		
Local Vector Table [local int 0]	11 0101	rw	
Local Vector Table [local int 1]	11 0110	rw	
Reserved	11 0111		
Initial Count Register	11 1000	rw	
Current Count Register	11 1001	r	
Reserved	11 1010		
Reserved	11 1011		
Reserved	11 1100		
Reserved	11 1101		
Divider Configuration Reg.	11 1110	rw	
Reserved	11 1111		

NOTE:

Address space 101000 to 101111 and 111111 are reserved

2

13.0 TIMING DIAGRAMS



13.0 TIMING DIAGRAMS (Continued)

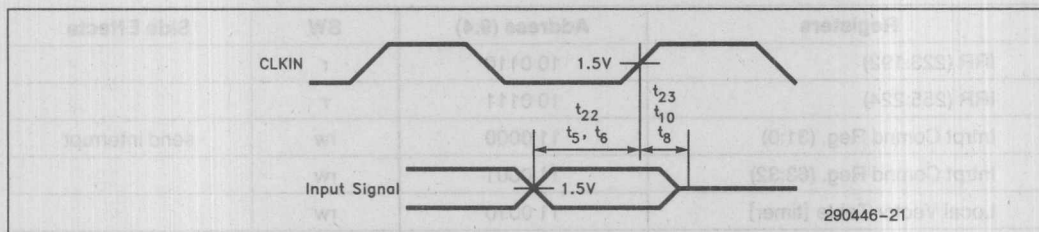


Figure 31. Input Waveforms

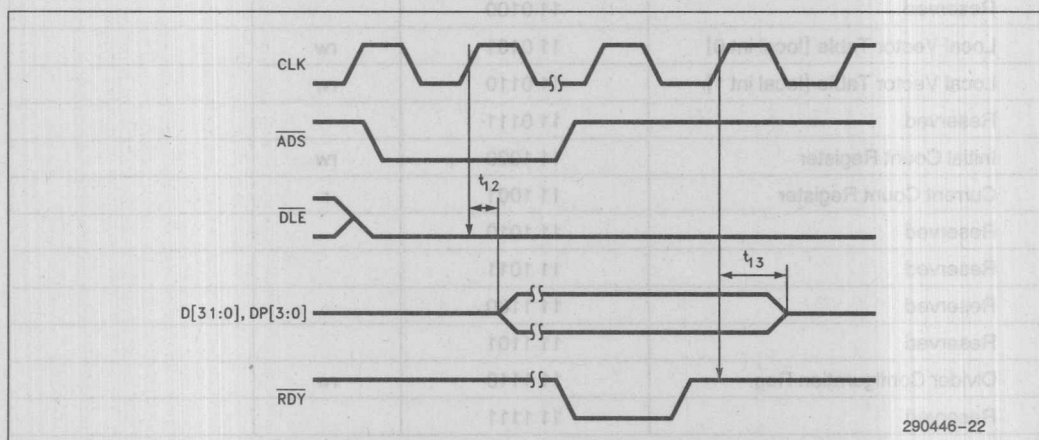
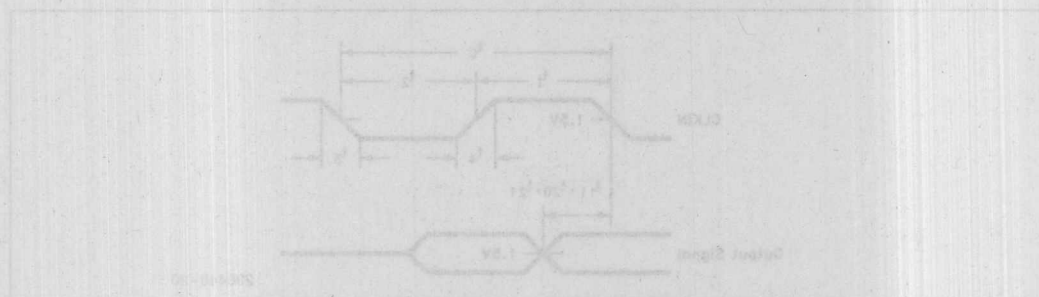
Figure 32. Data Bus Tri-State Delays when \overline{DLE} Sampled Low with \overline{ADS} 

Figure 33. Output Waveforms

13.0 TIMING DIAGRAMS (Continued)

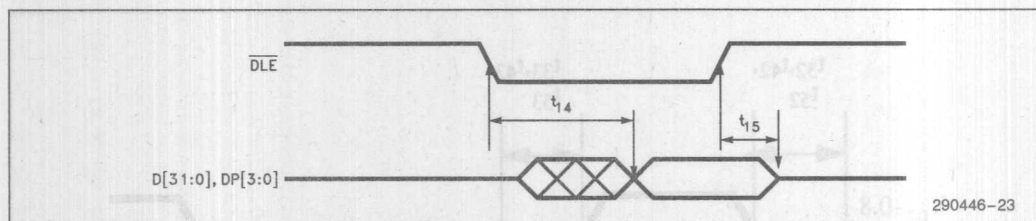


Figure 33. Data Enable/Disable Delay when DLE is Sampled High with ADS

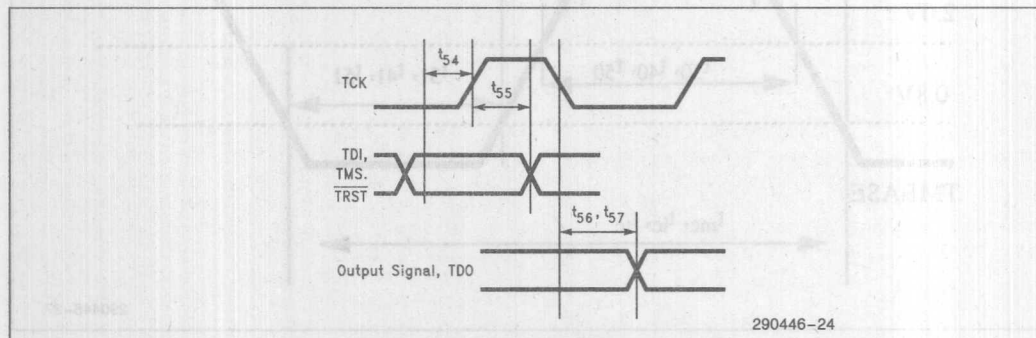


Figure 34. TAP Signal Timings

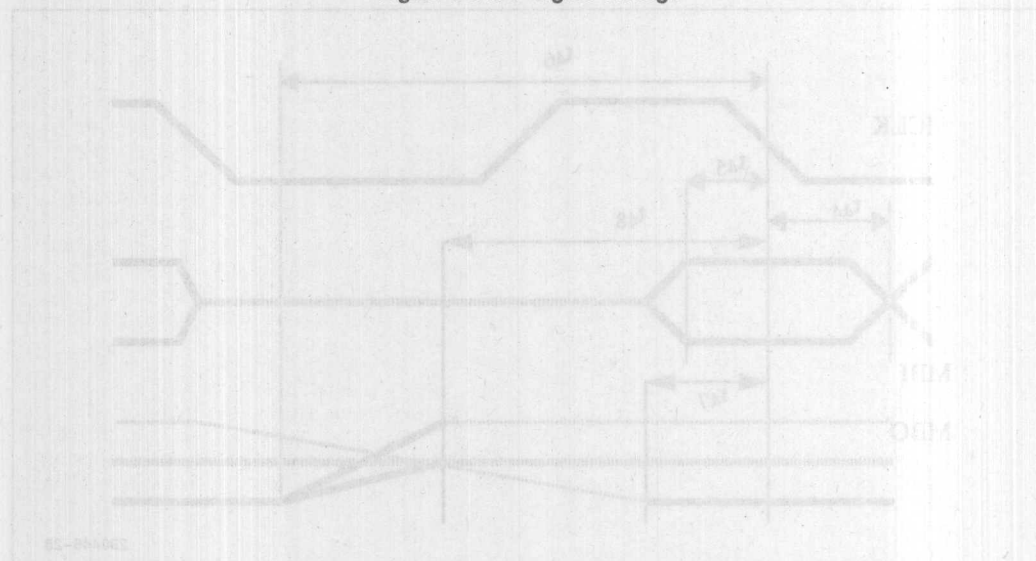


Figure 35. I/O Bus Open-Drain Output Delay

2

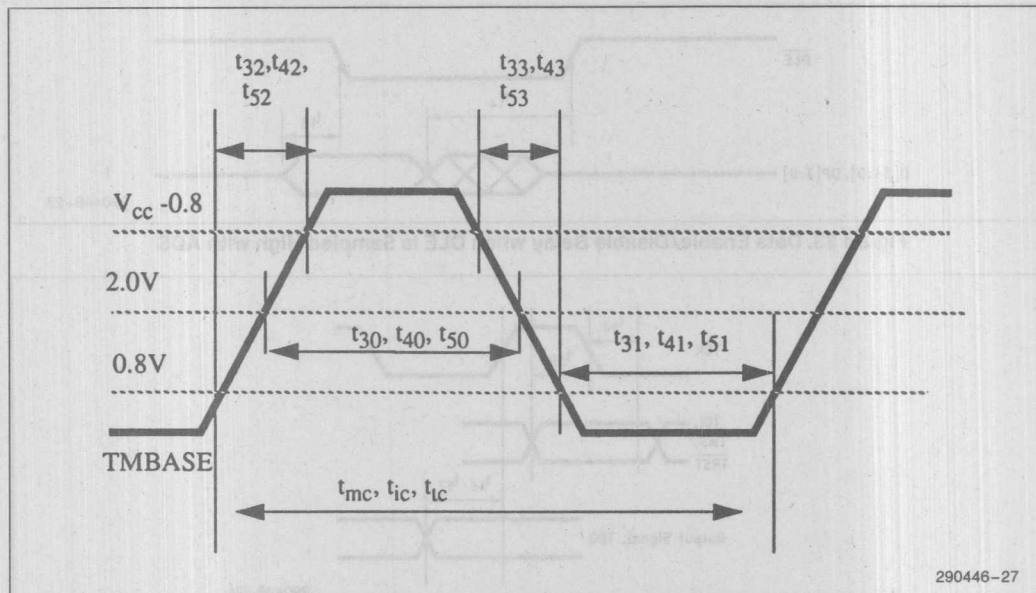


Figure 35. TMBASE, ICLK, TCK Timing

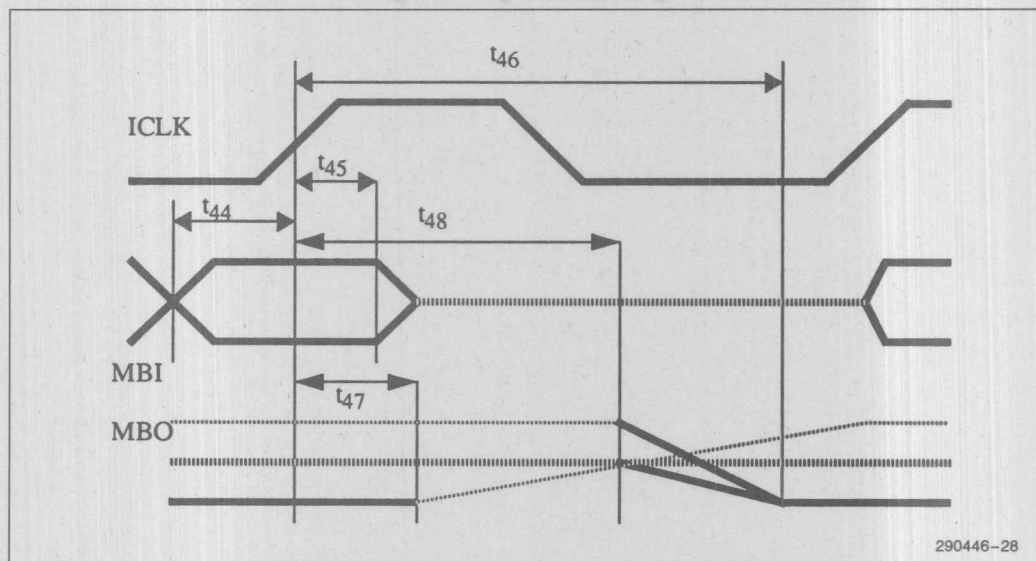
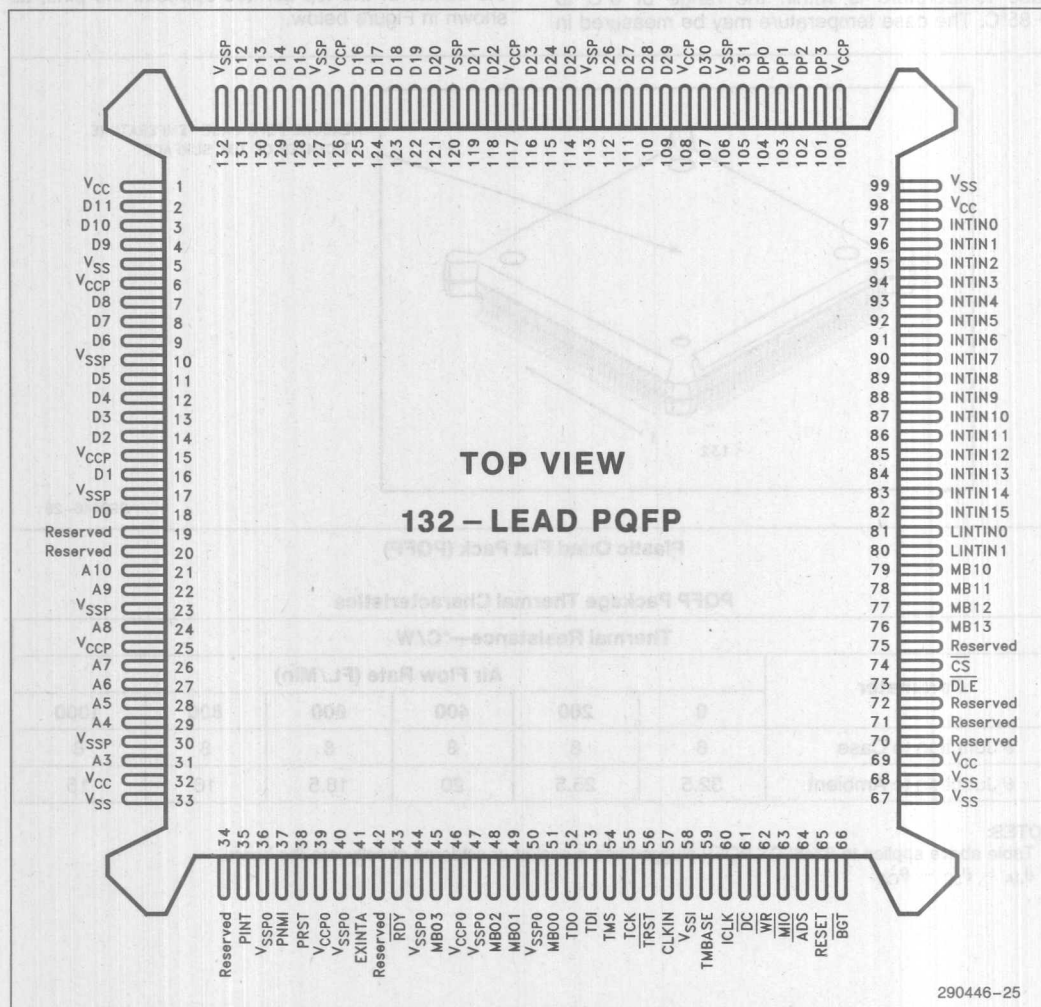


Figure 36. ICC BUS Open-Drain Output Delay

14.0 PACKAGE PIN-OUT

132-Lead PQFP

Package Type KU (See Packaging Specification, Order Number 240800)



NOTE:

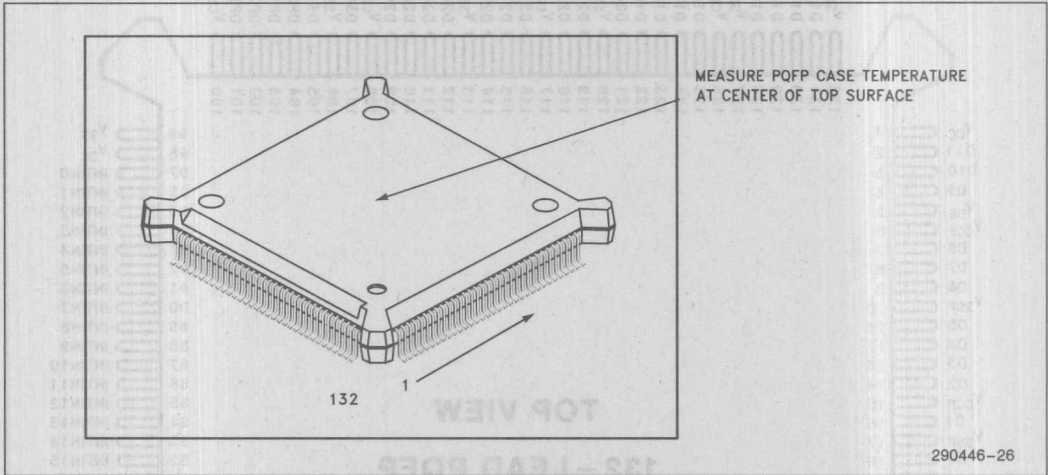
See pin description section for appropriate pin-strapping of the reserved pins.

15.0 PACKAGE THERMAL SPECIFICATION

The 82489DX is specified for operation when the case temperature is within the range of 0°C to +85°C. The case temperature may be measured in

any environment, to determine whether the device is within the specified operating range.

The PQFP case temperature should be measured at the center of the top surface opposite the pins, as shown in Figure below.



Plastic Quad Flat Pack (PQFP)

PQFP Package Thermal Characteristics

Thermal Resistance—°C/W						
Parameter	Air Flow Rate (Ft./Min)					
	0	200	400	600	800	1000
θ Junction to Case	8	8	8	8	8	8
θ Junction to Ambient	32.5	25.5	20	18.5	16	15

NOTES:

1. Table above applies to 82489DX PQFP plugged into a socket or soldered directly into the board.
2. $\theta_{JA} = \theta_{JC} + \theta_{CA}$.

16.0 GUIDELINES FOR 82489DX USERS

16.1 Initialization

This section outlines one possible initialization scenario. Other scenarios are certainly possible, and one would be selected as part of a platform standard initialization scheme. The intent of this section is to illustrate that the initialization support provided by the 82489DX is adequate to support MP (Multi-processor) system initialization.

Each 82489DX has a RESET input pin connected to a common Reset line. Upon system reset, this common reset line is activated, causing all the 82489DXs to go through reset. All 82489DX local units (note: only local units and not I/O units) latch their ID from their address bus on reset. The ID can be provided by the bus control agent based on slot number.

The local units next assert their processor's Reset pin, holding the processor in reset, and next perform their internal reset, setting all registers to their initial state. The initial state of all 82489DX Units (both local and I/O units) is "all masks set" and all Local Units disabled; registers are otherwise initialized to zero. Note that the PINT and PNMI output pins are in tri-state mode when the local unit is disabled. After this, each 82489DX local unit will deassert its processor's Reset pin, allowing the processors to come out of reset and perform self test and start executing initialization code.

Note that while connecting PRST pin it should be noted that whenever PRST pin is activated by 82489DX either because of software reset message or hardware reset, the 82489DX itself is reset. It should be taken care in the cases of Warm reset where only processors need to be reset and not the interrupt controller. In brief, the usage of PRST depends upon the system requirement on various reset.

Somewhere in this code sequence, the processors that are "alive" will enable their 82489DX local units, and attempt to force all the other processors back into Reset. Forcing the other processors into reset is performed by sending them the inter-processor interrupt with Destination Mode = "Physical", Delivery Mode = "Reset", Trigger Mode = "Level", Level = "1", and Destination Shorthand = "All Excl Self". Only the first processor to get the ICC bus will succeed in sending this signal and reset all other 82489DXs and their processors. The other processors are kept in reset until such time that an MP operating system decides they can become active again. The only running processor next performs the rest of system initialization.

Eventually, an MP operating system will be booted at which time the operating system would send "deassert reset" interprocessor signals to activate the other processors in the system. A mechanism must be provided by the platform that allows the added processors to differentiate the very first reset from a subsequent one.

16.2 Compatibility

COMPATIBILITY LEVELS

The 82489DX can be used in conjunction with standard 8259A-style interrupt controllers to provide a range of compatibility levels.

At the lowest level we have "PC shrink-wrap" compatibility. This level effectively creates a uniprocessor hardware environment within the MP platform capable of booting/running DOS shrinkwrap software. In this mode, only the 8259A generates interrupts and the 82489DX becomes a virtual wire. The interrupt latency can be minimized by connecting the 8259A interrupt to local unit directly.

The next level preserves the software compatible view of an 8259A but it allows more than one processor to be active in the system. This results in an asymmetrical arrangement, with one processor fielding all 8259A interrupts but with added inter-processor interrupt capability. In this mode, 82489DX "merges" 8259A interrupts with inter-processor interrupts. Existing I/O drivers would be bound to the compatible CPU and interface directly with the 8259A.

At the next compatibility level, 8259A compatible drivers can be mixed with native 82489DX drivers. Devices can generate interrupts at either 8259A or an 82489DX. This provides for partial symmetry as individual drivers migrate from the 8259A to native 82489DXs.

Another 8259A compatible point can be defined for MP systems. Each processor could have its own compatible 8259A controllers, allowing multiple processors to run compatible I/O drivers, but statically spreading the load across the available processors.

82489DX/8259A INTERACTION

The principle of compatible operation is very straightforward; the 82489DX(s) become a virtual wire connecting the 8259A's INT output through to the processor, while at the same time making 8259A visible to the processor.

The two connection schemes described only differ in the number of 82489DX(s) (one or two) that are located in the path from the 8259A to the processor. In the one 82489DX example illustrated in Figure 37, the INT output of the 8259A connects to one of the Interrupt Input pins of the 82489DX through an edge generation logic. This could be an interrupt pin on the 82489DX's I/O unit or local unit; assume a local interrupt input is used. The Local Vector Table entry for the interrupt pin that connects to the 8259A is set up with a Delivery Mode of "ExtINT" and edge trigger mode. This indicates that the interrupt is generated by an external controller. The processor's INT pin connects to the 82489DX PINT pin.

This setup enables the 82489DX local unit to detect assertions (up-edges) of the 8259A's INT output pin and pass this on to the processor's INT input. 82489DX asserts ExtINTA pin along with (one clock prior to) PINT pin to indicate "8259" interrupt. When the processor performs its INTA cycle the 82489DX itself does not respond other than deasserting PINT to the processor. At the third clock after ADS in the second bus cycle of INTA cycle ExtINTA is deasserted. External logic should make use of the ExtINTA signal to make the INTA cycle visible to the 8259A and the 8259A should provide the vector. At the same time, the local unit considers the external request as delivered, and need not wait for the external 8259A's INT to be deasserted. *A new up-edge must be generated on the 8259A INT pin before the local unit will assert the processor's INT pin on behalf of the 8259A. External edge generation logic should be used for this.* Compatible software interacts directly with the 8259A.

The mechanism is essentially the same in the two-82489DX scheme. The difference is that the 8259A connects to an interrupt input pin of the 82489DX I/O unit in the I/O system. The Redirection Table entry for this pin is again programmed with an "ExtINT" Delivery Mode, and the (single) 82489DX destination local ID corresponding to the compatible DOS processor. Capturing the up-edges of the 8259A's INT pin by the 82489DX local unit now involves sending messages from the 82489DX I/O unit to the 82489DX local unit via the ICC bus. The "virtual wire" now includes messages over the ICC bus.

Adding inter-processor ICC interrupts (or any other 82489DX generated interrupts) to the compatible operation is accomplished by having the 82489DX internally OR the 8259A's INT request with any 82489DX interrupt request.

Before the 82489DX actually sends the interrupt signal to the processor, the 82489DX decides whether it does this for an 82489DX interrupt or whether it does this on behalf of the external controller. When the processor performs the corresponding INTA cycle, only the 82489DX knows whether it should re-

spond with a vector, or whether the external 8259A should.

If the 82489DX needs to respond, then it will enable an externally implemented trap that prevents the 8259A from seeing the INTA cycle. If the 8259A needs to respond, then the 82489DX will not enable the INTA trap, and the INTA will be allowed to reach the 8259A. 82489DX implements this by asserting its EXTINTA pin to indicate external 8259A should respond with the vector. The 82489DX local unit controls the INTA trap via its "ExtINTA" output pin; the 82489DX does not actually provide the trap itself.

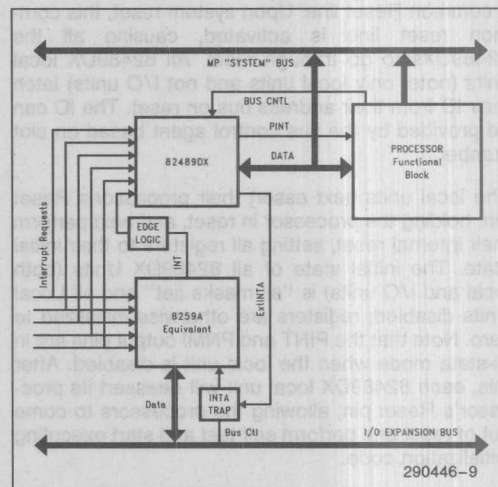


Figure 37. Edge Logic

82489DX/8259A DUAL MODE CONNECTION

In systems that can be booted either as a configuration with compatible 8259A or without, device interrupt lines are connected to both the Interrupt Request pins of the 8259A and Interrupt Input pins of the 82489DX with all interrupts either masked at the 82489DX or at the 8259A. Some EISA and Micro-Channel chip sets that include on-chip 8259As also have internally connected interrupt requests. For example, the 82357 (the ISP of the EISA chipset) generates timer and DMA chaining interrupts internally. These are not available as separate interrupts outside the ISP. In non compatible mode the ISP timers are not used, since each local 82489DX unit provides its own timer. Therefore, the ISP's 8259A is configured to mask out all interrupts except the DMA chaining interrupt which is configured in level-sensitive, auto EOI mode. This causes the 8259A's INT output to track the state of the internal DMA interrupt request. The 8259A's INT output is then connected to one of the 82489DX interrupt input pins programmed to generate a regular (i.e., not

"ExtINT") level-sensitive interrupt. The ISP 8259A then no longer functions as an external interrupt controller; it has been logically disabled, and it needs no interrupt acknowledge or EOI. The INTA and EOI cycles occur only at the 82489DX. It should be noted that 82489DX accepts only active high level/edge interrupt inputs. External programmable logic should take care of polarity reversal that may be needed in EISA system for sharing of interrupts.

16.3 Hardware Guidelines

82489DX HARDWARE STATE ON RESET

The 82489DX goes to reset state either by Hardware Reset state or Software Reset message received on the ICC bus. On reset, 82489DX is disabled. The following is the hardware state of 82489DX after reset.

PRST	Active (HIGH)
PNMI	TRI-STATED (Internal Pull-Down Provided)
PINT	TRI-STATED (Internal Pull-Down Provided)

82489DX is disabled on Reset and unless specifically enabled, it does not start its interrupt mechanism.

The difference between hardware reset and software reset message is that during hardware reset 82489DX samples the address bus and stores the last sample in Local Unit ID whereas for software reset it does not sample and store the unit ID. In addition, during the hardware reset pulse should be wide enough to accommodate for at least one rising and falling edge of ICLK. On hardware reset ExtINTA is held high.

PULL UP AND PULL DOWN RESISTORS

PNMI, PINT are tri-stated at power on and they are maintained in tri-state condition till the unit is enabled. Even though internal pull down resistor is provided on PNMI and PINT external additional pull down resistor may be needed depending upon the loading on these pins by external logic. The DC characteristics gives the control specification from which the value of resistor, if needed, can be calculated.

It should be kept in mind that the ICC bus being electrically open drain bus requires pull up resistors at the MBO pins. ICC bus output low current is just 4 mA.

PINT and ExtINTA Timings

It should be noted that for ExtINTA type of interrupts **PINT gets activated one clock after ExtINTA gets activated**. When getting deactivated, both PINT and

ExtINTA gets deactivated at the same clock edge.

ExtINTA Timings

In the interrupt acknowledge cycle for External Interrupt control, 82489DX asserts ExtINTA. It decodes the type of cycle from CPU control signals like M/I \bar{O} , D/ \bar{C} and W/ \bar{R} . CPU does two bus cycles back to back for interrupt acknowledge cycle. 82489DX maintains ExtINTA active throughout the first cycle. For next cycle (when the vector will be given by external 8259) after 82489DX senses the start of the cycle (by ADS) 82489DX deactivates ExtINTA. External control logic may be inserting wait states to match the 8259 timings. Since 82489DX has no way of finding out the cycle completion, 82489DX deasserts ExtINTA before the second bus cycle gets completed. This should be kept in mind while using ExtINTA for external interrupt control logic.

82489DX AND MEMORY MAPPING

The 82489DX is a 32-bit high performance interrupt controller. It allows the CPU to do 32-bit read and write to it. By memory mapping 82489DX the system performance can be enhanced. It should be noted that 82489DX does not support pipelining. Even though 82489DX can be memory mapped, its functionality as an interrupt controller should be kept in mind while programming the virtual memory management control data structure. The caching policy for the page where 82489DX is mapped should also be done with the functionality of 82489DX in mind. For example, the reads to 82489DX should not be cached and writes should be write-through. Since 82489DX registers are aligned at 128-bit boundaries, memory mapping 82489DX with interleaved memory system should not be a problem.

JTAG CIRCUIT CONSIDERATIONS

The JTAG circuit is used for boundary scan test. The JTAG pins has a TCK, (JTAG clock), TRST, (JTAG Reset), TDI, (Test Data Input), TMS, (Test Mode Select) and TDO, (Test Data Output).

The JTAG circuitry, if not used, should be properly deactivated so that it will not interfere in the normal functional operations. The JTAG can be inactivated in any one of the following ways:

1. JTAG inactivation through TRST: The TMS, Test mode Select should be either left open (internal pull up is provided) or tied to V_{CC}. The TRST can be pulsed low (bring it low and after meeting the pulse width requirement bring it back high again) to keep the JTAG circuitry to idle state. The TRST pulse brings the JTAG circuitry to idle state and TMS being kept high maintains the JTAG circuitry in idle state.

2. JTAG inactivation through TCK: The TMS, Test mode select should be either left open (internal pull up is provided) or tied to V_{CC} . The TCK within 5 clocks brings the JTAG circuitry to idle state. The TMS, being held at logic high level, maintains the JTAG circuitry in idle state.

16.4 Programming Guidelines

The 82489DX register data structure contains different fields to specify the mode of operations and the options available within each mode. Since certain options are applicable to specific modes only (for example "Remote Read" mode applies only to interrupt command register, it does not have relevance to I/O unit's redirection tables) the following programming guidelines are provided.

UNIQUE ID REQUIREMENT

All the local units and I/O units hooked in a ICC bus should have unique ID before they can use the bus. This should be ensured by the programmer since for ICC bus arbitration the units (whether it is local unit or I/O unit) arbitrate with their unit ID.

For future compatibility, the Units should be assigned IDs starting with 0, 1, 2 etc. with the highest ID in the system being number of units minus 1. So in a four 82489DX system there are four local units and four I/O units. The ID starts with 0 and the highest ID in the system will be 7. Note that each unit should have different ID in the system.

ATOMIC WRITE READ TO TASK PRIORITY REGISTER

Normally, the task priority register is written with highest priority to mask certain low level interrupts before entering into critical section code. In a system where 82489DX is memory mapped the CPU may buffer this task priority register write to its on chip write buffer. The following scenario can happen in such situation: CPU posts task priority register write to its on chip write buffer and enters into the critical code. A lower priority interrupt (which should not enter the critical code) interrupts the CPU before the write buffer gets flushed into task priority register. The CPU accepts the lower priority interrupt. To avoid the situation atomic write read to task priority register should be done. The read following write ensures that the write buffer is flushed to task priority register and the atomicity ensures that no interrupt will be accepted by the CPU during its write to task priority.

It should be noted that if the CPU does interrupt acknowledge cycle only after flushing the write buffers then the above situation may not arise.

CRITICAL REGIONS AND MUTUAL EXCLUSION

Each 82489DX has a single Interrupt Command Register that it uses to send interrupts to other processors. The programmer should make sure to synchronize access to this register. Specifically, 1). writing all fields of the register, 2). Sending the interrupt message (by writing the LSB register), and 3). waiting for Delivery State to become Idle again, should occur as a single atomic operation. For example, if interrupt handlers are allowed to send inter-processor interrupts, then interrupt dispensing to the processor must be disabled for the duration of these activities.

INTERRUPT COMMAND REGISTER PROGRAMMING SEQUENCE

The interrupt command register (31:0) has a side effect of sending interrupt once it is written. The destination is provided in the interrupt command register (63:32). So always interrupt command register (63:32) should be programmed before programming interrupt command register(31:0).

Program Interrupt Command Register (63:32)



Program Interrupt Command Register (31:0)

INTERRUPT VECTOR

Two different interrupts should not be programmed with the same interrupt vector.

LOCAL AND I/O UNIT

Only Interrupt command register supports "Remote Read" Delivery mode. Local and I/O unit interrupts do not support "Remote Read".

ICR (INTERRUPT COMMAND REGISTER)

1. ExtINTA delivery mode is not supported for all destination shorthands.
2. "Remote Read" should always be programmed as "Edge" triggered interrupt.
3. "Remote Read" should always be programmed with physical destination mode (and not with Logical Destination mode). Broadcast addressing should not be used for Remote Read.
4. For "all incl Self" and "All exc. self" destination shorthands, "remote read" delivery mode should not be used.
5. For "all incl self" and "self" destination shorthands "Reset" delivery mode should not be used.

ICR (INTERRUPT COMMAND REGISTER)

(Continued)

6. For "all exc self" destination shorthand if "Reset" delivery mode is used, it should be ensured at system level that only one processor executes this instruction at any time.
7. Messages could be sent out in "Logical" or "Physical" mode with destination ID of all 1's depending on the way Destination mode entry is programmed. In brief, "All incl self" and "All exc. Self" support both "Logical" and "Physical" addressing mode.
8. When destination shorthand (i.e., broadcast) is used with lowest priority destination mode, then even though all participates in arbitrating for destination, only the lowest priority gets the message. So even though the addressing is broadcast since the destination mode is lowest priority only one gets the message.
9. When destination shorthand (i.e., broadcast) is used with "Fixed" destination mode, then all the units get the message.

ISR/IRR/TMR

Bits 0–15 of IRR/ISR/TMR do not track interrupt. No interrupt of vector numbers from 0–15 can be posted. The total interrupt supported are 240. When reading the lowest 32 bits of these registers, 0 will be returned for the lower 16 bits.

FOCUS PROCESSOR

Focus processor is applicable only within the addressed units.

ExtINT Interrupt Posting

The external interrupt has no priority relationship with the 82489DX priority. But when posting an interrupt to the processor, if both an external interrupt and a 82489DX interrupt are pending, 82489DX could post either one to the processor. In 82489DX implementation, it would post external interrupt whenever there is no other 82489DX interrupt that can be posted to the processor. It should be also noted that External interrupts can not be masked by raising task priority. However, they can be masked by the mask bit in the table entry for the ("ExtINT") external interrupt.

The extINT interrupts are specific in their characteristics in that they do not have any priority relationship with the rest of the interrupt structure. ISR and IRR bits in 82489DX are used to do the housekeeping functions for interrupt priority. Since extINT interrupts do not have any priority relationship, ISR and

IRR bits are not maintained for external interrupts. As far as interrupt acceptance is concerned, if more than one extINT interrupts are directed towards a local unit, that local unit treats all the extINT interrupts directed to it as only one extINT interrupt. This leads to an important point that in a system not more than one interrupt should be programmed as extINT interrupt type with the same destination. It should be noted that there can be more than one extINT type of interrupt in a system with each having different local unit as destination.

Synchronizing Arb IDs

Initialization of an 82489DX's local unit ID is implementation dependent. In some platforms, power-on reset will latch the right values into the 82489DXs; in other platforms, unique IDs may be assigned by initialization firmware. In both cases the 82489DX I/O unit should be assigned unique ID by initialization firmware. The important point is that the 82489DXs are required to have unique IDs before they can use the bus, and in addition, all their Arb IDs must be "in sync". Synchronizing Arb IDs is accomplished as a side effect of a "deassert reset" interrupt command. This resets the (rotating) Arb ID to the (constant) unit ID; it assumes that all 82489DXs have their unique ID.

LOWEST PRIORITY

"Only once delivery" semantics for a group destination is guaranteed only if multiple fixed delivery of the same interrupt vector are not mixed. For lowest priority arbitration to work, all the arbitration ID of local 82489DXs in the system should be in sync. This means after local unit IDs are written in all local units (each ID should be different from other IDs) a RESET DEASSERT message should be sent in ALL INCLUSIVE mode. The RESET DEASSERT message should be sent before system is used for lowest priority arbitration. This ensures that all ARB IDs are also different. (Arb IDs are copied from local unit IDs during RESET DEASSERT message.)

The RESET DEASSERT message, if not sent, only one delivery semantics may not be guaranteed in the cases where lowest arbitration is used in the system.

DISABLING LOCAL UNIT

Once the 82489DX is enabled by setting bit 8 of spurious vector register to 1, the user should not disable the local unit by resetting the bit to 0. The result will put the local unit in an inconsistent state. The local unit can be disabled by sending "reset" interrupt message to the local unit.

ISSUING EOI

EOI, End of Interrupt issuing indicates end of service routine to 82489DX. The ISR bit which is set during INTA cycle gets cleared by EOI. This section discusses the relevance of EOI to the specific types of interrupts and its timing related to interrupt deassertion.

EXTERNAL INTERRUPTS AND EOI

External Interrupts should be programmed as edge type. INTA cycles to external interrupts are taken automatically as EOI by 82489DX. This is similar to AEOI, Automatic End of Interrupt of 8259A. So there is no need to issue EOI to 82489DX for external interrupt servicing. This is done to achieve software transparency in the compatible mode.

SPURIOUS INTERRUPTS AND EOI

Spurious interrupts do not have any priority relationship to other interrupts in the system. So IRR is not set for spurious interrupts. EOI should not be issued for spurious interrupts. It is advisable not to share the spurious interrupt with any vector. If spurious interrupt vector is shared with some other interrupt then while servicing issuing EOI depends on the source of interrupt. If the source is spurious interrupt (for which the corresponding IRR is not set) then EOI should not be used. If the source is a valid interrupt sharing the spurious interrupt vector (for which the IRR is set) then EOI should be issued.

NM AND EOI

For NM type of interrupt no IRR bit is set. So EOI should not be issued while servicing NMI type of interrupts.

TASK PRIORITY REGISTER

Task priority register is used to specify the priority of the task the processor is executing. In 8259 the priority is defined only among the interrupts that it handles. 82489DX goes farther ahead in handling priority. In multitasking system, in addition to device interrupts various tasks have different priority and 82489DX allows consideration of the priority at system level. The processor specifies the priority of the task it executes by writing to task priority register. Now any interrupts at and below the task priority will be masked temporarily till the task priority gets lowered. The masking granularity is at priority level. Out

of 256 interrupt vectors 16 priority levels are specified and 16 vectors share one priority level. Since the masking granularity by the task priority register is at priority level, group of 16 vectors get masked when task priority register is increased by one level.

When task priority is at its minimum level of 0, interrupt vectors having level 1 to 16 are passed to CPU. Stated in other words, even when the task priority register is at its minimum (of level 0), interrupt vectors at level 0 will be masked. This means that the interrupt should not be programmed with vectors 0 to 15. So out of 256 interrupt vectors, only 240 interrupt vectors (vector 16 to 255) can be used in 82489DX.

ExtINT INTERRUPT AND TASK PRIORITY

ExtINT interrupt does not have any priority relationship with other interrupts or task priority register. So ExtINT interrupt can not be masked by raising task priority. They can be masked by writing to the vector table entry which corresponds to ExtINT interrupt.

REMOVING MASKS

When enabling units and removing Mask bits in situations where a device may already be injecting interrupts into the 82489DX system, the Mask in the Redirection Table should be removed last to ensure proper initial state (e.g., Remote IRR bit matching IRR in local unit).

DELIVERY MODE AND TRIGGER MODE

It is software's responsibility to make sure that Delivery Mode and Trigger Mode are set to meaningful combinations as listed below.

Delivery Mode	Trigger Mode
Fixed	edge/level
Lowest Priority	edge/level
Remote Read	edge
NMI	level
Reset	level
ExtINT	edge

Software is also responsible for not using meaningless Delivery Modes in Redirection Table entries and local Vector Table entries (e.g., use of Remote Read delivery mode).

ASSIGNING INTERRUPT VECTORS

Software has total control over the assignment of interrupt vectors to interrupt sources. The operating system writer should be aware of a number of things when doing this assignment.

Some processor architectures assign a predefined meaning to some of the vectors (i.e., entries in the interrupt table) as entry points to certain trap and exception handlers (e.g., divide error, invalid opcode, page fault, etc.). The programmer is strongly advised not to reuse these vectors.

The programmer must also be careful when using the same vector number to represent different interrupt sources (sharing vectors). This is especially true for level triggered interrupts. When multiple sources with different Redirection table entries share an interrupt vector, any of the sources deactivating its level signal will remove the interrupt request for all sources. Giving each interrupt source its interrupt vector in any case is the preferred approach.

SENDING INTER-PROCESSOR INTERRUPTS

Each 82489DX has a single Interrupt Command Register that it uses to send interrupts to other processors. It is software's responsibility to synchronize access to this register. Specifically, 1) writing all fields of the register, 2) sending the interrupt message (by writing the low register), and 3) waiting for Delivery State to become Idle again, should occur as a single atomic operation. For example, if interrupt handlers are allowed to send inter-processor interrupts, then interrupt dispensing to the processor must be disabled for the duration of these activities.

DELAY WITH LEVEL TRIGGERED INTERRUPTS

When a level triggered interrupt source deasserts its interrupt input, the destination will clear the interrupt's IRR bit only after receiving the message from the ICC bus. This introduces a small delay between the removal of the interrupt at the source and the removal of the interrupt at the processor. To avoid generating unnecessary interrupts, the interrupt handler should remove the interrupt at the source (at the device) as early as possible in the handler.

In any case, handlers should be able to deal with unnecessary interrupts.

RESET DEASSERT

A side effect of a reset deassertion message broadcast in the ICC bus is that all 82489DX local units reset their Arb ID to their unit ID. Interrupt com-

mands that use the "self" Destination Shorthand do not generate a message on the ICC bus. If software only wants to generate the side effect of resetting Arb IDs, it should use a command with Logical Destination Mode and a Destination field containing all zeroes.

INTERRUPT MASKING

There are a number of levels at which interrupts can be masked, each resulting in a different behavior on interrupt delivery.

- First, interrupt injection (or deliver) can be masked by setting the Mask bit in the interrupt's Redirection Table or local Vector Table entry. These interrupts are ignored, no message is sent for them. Granularity is an individual interrupt.
- Second, each 82489DX can individually mask interrupt dispensing by raising its Task Priority to some level. This 82489DX will not dispense interrupts to its processor of this and lower priority unless it is currently the focus of the interrupt. Note again that the 82489DX is designed to operate as fully nested with non-specific EOI (to use existing 8259A terminology). There is no explicit interrupt mask (such as MR) and there is no notion of specific EOI.
- Third, each processor may provide a mechanism that masks all interrupt dispensing to it using the processor supplied instructions or status bits to do so. This does not interfere with lowest priority arbitration of the processor's 82489DX local unit.

CHANGING REDIRECTION TABLES

Redirection Tables are typically set up at initialization time. When modifying a Redirection Table entry "on the fly" the programmer must be aware of state kept at other 82489DXs relative to the interrupt being modified.

DEVICE DRIVERS WITH 82489DX

It is strongly recommended to read the device status registers before servicing the device. This is because if an edge triggered device deasserts its interrupt before interrupt acknowledge cycle (it should NOT) 82489DX will NOT give spurious vector. It will give genuine interrupt vector corresponding to the device. So, interrupt service routine should validate the interrupt request before servicing the device.

SYSTEM HARDWARE AND SOFTWARE DESIGN CONSIDERATIONS

Design Consideration 1

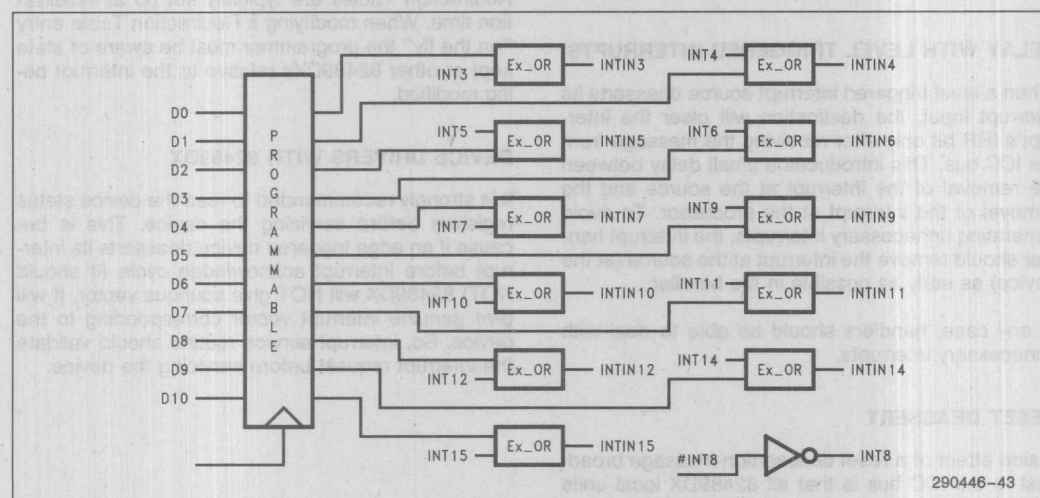
Description: The following design consideration has to be taken care of when using ISP (82357) as external interrupt controller. 82489DX allows connecting external 8259 type interrupt controller at one of its inputs. The mode associated with the interrupt input which has 8259 connected to it is called ExtINTA mode. 82489DX allows only EDGE TRIGGERED programming option for ExtINTA mode. But in the case of 82357, the INT output from ISP stays high in case more than one interrupt is pending at its inputs. It does not always inactivate its INT output after INTA cycle. This will lead to a situation where ISP keeps the interrupt at high level continuously and waits for INTA cycle. But since 82489DX expects an edge for interrupt sensing (for ExtINTA interrupts) it does not pass the interrupt to CPU and further interrupts are lost. So External circuitry should monitor the end of SECOND CYCLE of INTA cycle and force an inactive state at 82489DX's input. To avoid glitches at 82489DX input, this external logic should clear its output only at the end of second INTA cycle. It should be set by high going 82357 output. It should never be cleared by low going 82357 output. That is it should not follow 82357 output.

Design Consideration 2

Description: The following design consideration has to be taken care of when using 82489DX in EISA systems. EISA ISP(82357) chip integrates 8259A. It

additionally allows sharing of interrupts. To facilitate this sharing it has a programmable register, ELCR (Edge / Level trigger control register) by which certain interrupt inputs can be programmed as edge (low to high except for RTC) or level (the level is active low). The determination of edge or level is done during initial configuration of EISA system by reading EISA add in boards from the interrupt description data structures. The solution is to have programmable logic at the interrupt inputs so that 82489DX is compatible with EISA ISP. This will introduce one more register and logic to support this. This should be an 11-bit programmable register and an array of ExOR logic (12 ExOR gates or equivalent PLD). The ISP allows programmability of the following interrupts.

INT3 INT4 INT5 INT6 INT7 INT9 INT10 INT11 INT12 INT14 INT15. In addition to the above 11 interrupts, it fixes **INT8** to be active low edge triggered interrupt. INT8 is the only case where it is active low edge triggered type. So the following logic can be used to add programmability in 82489DX based EISA system. Before connecting these 11 interrupt lines directly (#INT8 which is from Real Time Clock is always active low edge triggered. #INT8 can be passed through an inverter since there is no need for programmability) to the 82489DX they should pass through an array of 11 Ex_OR gates. One input of Ex_OR gate connects to the corresponding INT pin and other input connects to a bit of programmable register. The output of Ex_OR gate is connected to 82489DX. The idea of Ex_OR is to use as a controlled inverter.

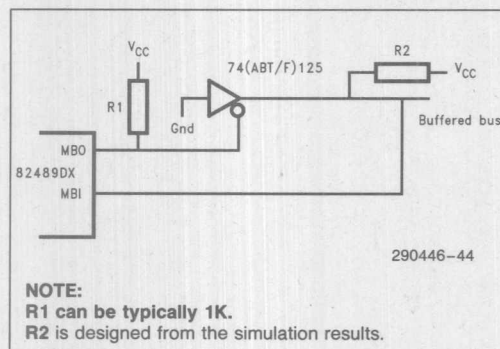


INTIN are the interrupt inputs to the 82489DX and INT are the system interrupt. The Ex_OR gating register is programmed after EISA configuration is found from add in boards as how these interrupt lines are going to be used in that particular configuration. If a particular input is edge triggered, then the corresponding bit in the register is written with 0. If a particular input is level triggered, then the corresponding bit in the register is written with 1.

8259 by itself does not have polarity control whereas 8259 when implemented in EISA chipsets have the polarity control. Similarly APIC does not have by itself polarity register. So polarity register should be programmed as a part of system BIOS and not APIC BIOS.

Design Consideration 3

I_{CC} bus drive is an open drain bus with drive capacity of 4 mA only. Since data is transmitted at each I_{CC} clock, the "charging" of I_{CC} bus should be fast enough to ensure proper logic level at each clock edge. The I_{CC} bus needs pull up resistors since it is open drain bus. Since the drive is only 4 mA, the pull up resistor value can not be less than 5V/4 mA. This being the limit of the resistor value, the length and the characteristics of the I_{CC} trace forces a capacitance value. Both the resistor and capacitance brings a RC time constant to the I_{CC} bus waveform. So, Electrical consideration has to be given to and practice of controlled impedance should be exercised for layout of the I_{CC} bus. The length of the trace should be kept as minimum as possible. If the length of the I_{CC} bus can't be kept less, than say 6 inch, because of mechanical design of the system, the external line drivers should be added to I_{CC} bus and I_{CC} bus should be simulated with the added driver characteristics.



Design Consideration 4

This is related to ADS#, BGT# and CS# timings. For bus cycles not intended for 82489DX, (CS# = 1 where 82489DX is supposed to sample it), any change in CS# line while the ADS# is still active, may erroneously cause a RDY# returned from 82489DX. Anomolous behavior may result if for BGT# ties low cases

- a) BGT# goes away just one clock after ADS# or
- b) ADS# is still active, and CS# changes during this period.

For other cases anomolous behavior results if CS# changes when ADS# is still active. The following considerations are important from timing point of view. Always limit the pulse width of 82489DX ADS# to one CLKIN. Also avoid changing levels on BGT#/CS# line, when ADS# is active for cases being identified as BGT# tied low (BGT# sampled low when ADS# goes active). Also avoid changing levels on CS# line when BGT# is active.

Design Consideration 5

82489DX does not recognize the interrupt when an edge occurs at the interrupt input pin while interrupt is masked. When later it is unmasked there is no further edge and so 82489DX never passes that forgotten edge and that interrupt channel becomes unusable after that.

The recommendation is that first 82489DX should be unmasked and then the device interrupt should be enabled in the device register. By this, software can ensure that always an edge will occur after an interrupt is unmasked.

Design Consideration 6

Description: Edge triggered interrupts should not deassert their output till they are acknowledged by INTA cycle from CPU.

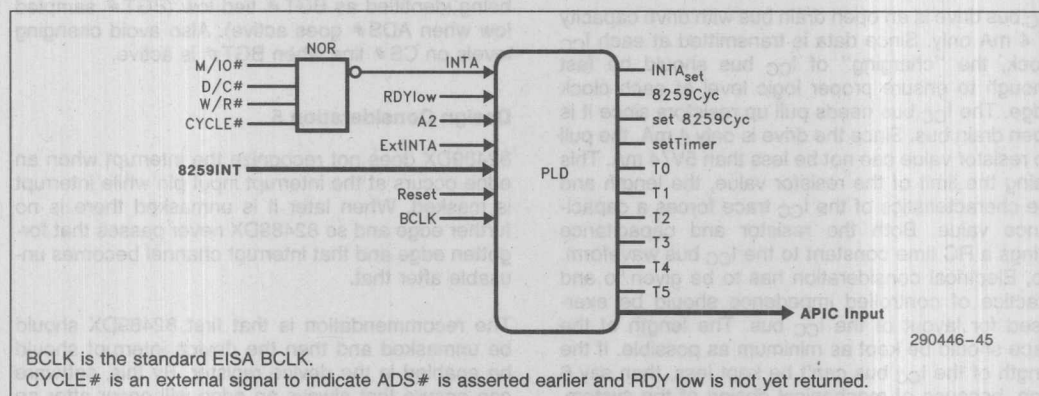
Issue:

82489DX employs glitch detection logic for edge triggered logic. To make sure the detected edge interrupt is not a glitch, 82489DX samples the input again before sending the interrupt message. The time difference between the first sampling of interrupt to be active and second sampling (just before sending the interrupt) is not a constant number. This is because the I_{CC} bus might have been occupied by other messages. So, for example if during first sampling it was detected that INTIN0 and INTIN15 are both active and after sending INTIN0 it samples INTIN15 again before sending message for

INTIN15. But between this time ICC bus might have been occupied by other messages. So even if an edge triggered interrupt is held active high for a really long time and then brought low before INTA cycle, it is considered as a glitch. Because it may happen that the second sampling occurred just when the interrupt line got low.

Once the glitch detection circuitry found this "glitch", it goes back to the state where it will start sampling and waiting for an active edge to occur. This takes more than one clock cycle (CLK) and if the "glitch interrupt" generates an edge before that time after the second sampling of low level is done, then the edge is lost forever.

Since the time when the second sampling is done is unknown, the best way is to make sure the edge triggered interrupts do not deassert their outputs till they are acknowledged by INTA cycle from CPU. It is found that in some cases 8259 can generate brief active low pulses on its output. So the glue logic between 8259 and 82489DX input pin should make sure that 82489DX input pin is clear only after getting second interrupt acknowledge cycle. The glue logic should not just follow the 8259 output. Put in other words, after interrupt acknowledge cycle to 8259, if the 8259 input is seen active high, it should generate an edge at 82489DX input. Moreover, even if 8259 output goes low the glue logic should not lower its output since the only time when the glue logic can deassert its output is when it finds an interrupt acknowledge cycle for 8259. The following PLD equations and schematics serves as an example for the glue logic between 8259 and 82489DX.



APIC input = $\text{/T0} * \text{/Reset} * \text{APIC input} * 8259\text{INT} * \text{INTA}_{\text{set}}$; Sample 8259 interrupt
 $+ \text{/T0} * \text{/Reset} * \text{APIC input} * \text{INTA}_{\text{set}}$; Hold till it is cleared by delayed interrupt acknowledge
 $+ \text{/INTA}_{\text{set}} * 8259\text{INT}$

Set 8259Cyc = $\text{/Reset} * \text{INTA} * \text{ExtINTA}$; This INTA cycle is for 8259

8259Cyc = $\text{set8259Cyc} * \text{/T0} * \text{/Reset}$; Set 8259cyc will set 8259 cycle and T0 will clear it
 $+ \text{/T0} * \text{/Set8259Cyc} * 8259\text{Cyc} * \text{/Reset}$; Hold 8259 cycle till T0 clears it

INTA_{set} = $\text{/Reset} * \text{/INTA}_{\text{set}} * \text{INTA}$; wait for very first INTA cycle after reset
 $+ \text{/Reset} * \text{INTA}_{\text{set}}$; once first INTA cycle after Reset is found, set the INTA_{set}

Set timer = $8259\text{Cyc} * \text{/A2} * \text{/RDYlow}$; Start the timer at end of second INTA cycle

T0 = $\text{Set timer} * \text{/T5} * \text{/Reset}$; Set timer will set T0 and T5 will clear
 $+ \text{/Set timer} * \text{T0} * \text{/T5} * \text{/Reset}$; Till T5 clears it hold T0

T1 := $\text{T0} * \text{/Reset} * \text{/T5}$; Follow T0 after one clock for setting but clear along with T0

T2 := $\text{T1} * \text{/Reset} * \text{/T5}$; Follow T0 after two clock for setting but clear along with T0

T3 := $\text{T2} * \text{/Reset} * \text{/T5}$; Follow T0 after three clock for setting but clear along with T0

T4 := $\text{T3} * \text{/Reset} * \text{/T5}$; Follow T0 after four clock for setting but clear along with T0

T5 := $\text{T4} * \text{/Reset}$; Follow T0 after 5 clock for setting

290446-46

NOTES:

T1, T2, T3, T4 and T5 are clocked signals and others are combinatorials.

This circuit and PLD equations are given for concept clarification purpose. They are not tested.

INTA_{set} is needed so that some 8259 logic at power on activates its INT output to 1 and it deactivates its output after only 8259 initialization (which should happen after APIC initialization) and since APIC needs to detect rising edge at 8259, it is essential to follow the 8259 until first interrupt. This is the only occasion 8259 output will be just followed.

DIRECTIONS FOR EASY MIGRATION TO FUTURE INTEGRATED APIC

The following are the software programming directions Intel strongly recommends for easy migration from 82489DX to integrated APIC. The audience to this portion of the document are both hardware designers and firmware developers for APIC based systems. In the following discussions, the APIC BIOS is viewed functionally as two subsections 1) APIC BIOS which are all interrupt vector, priority, interrupt distribution related functions and the remaining portion of BIOS which is referred to part of system BIOS which is responsible for interrupt polarity programming, starting next processor, etc.

Note that the names APIC BIOS and APIC DRIVER are interchangeably used in the following discussion. Different Operating systems refer such functional module differently.

Consideration 1

Question: The logical destination register in future implemented APIC may have only 8 MSBs defined and 82489DX has 32 bits specified. Will this hinder binary level compatibility?

mode) 82489DX can go up to 32 CPUs whereas future APIC can go up to 8 CPUs with flat logical addressing mode. **For binary compatibility, it is strongly recommended that 82489DX software use ONLY 8 MSB of logical destination register.**

Consideration 2

Question: The present day MP systems with external control ports for starting next processor may program those external control ports for starting next processor. APIC DRIVER may use external control ports for starting next processor. In future implementations of APIC, the starting of next processors may use more refined mechanisms which may not use external control ports. Will this introduce compatibility problem?

Response: Again, the starting of next processor is really part of MP system DRIVER and depending of the mechanism used to start next processor it will vary. In future implementation if starting next processor is done using new mechanisms, the starting next processor portion of MP DRIVER will be changed accordingly. Even though this will not result in any change in the APIC DRIVER which deals with interrupt priority, distribution, etc., the corresponding change will be needed in the starting application processors portion of DRIVER.

One possible method of implementing software is using version register. Version register is different in 489DX and future implementations of APIC. Taking care of these differences, such as mechanism for starting next processor, should be possible using version register.

Consideration 3

Question: APIC architecture, by its nature, seems to misinterpret spurious interrupts as genuine interrupts. That is, if an edge triggered interrupt goes inactive before interrupt acknowledge cycle, APIC, instead of giving spurious interrupt vector, gives genuine interrupt vector. Is it true that this is not the case with 8259? If that is the case, drivers which do not check device status registers for servicing the device may work with 8259 but may not work with APIC. Is this a compatibility problem?

Response: No, this is not true. Even with 8259 there is a time window in which a similar thing can happen. For example if interrupt goes inactive just after first INTA cycle but before second INTA cycle 8259 will also signal this spurious interrupt as genuine interrupt. So drivers which do not check device status registers may also fail with 8259.

Our strong recommendation to device drivers is to read device status register before servicing the device. If the device status register indicates that there is no valid source of the interrupt, the service routine should just issue EOI and return. It should not service the device. This should take care of the new drivers that will be written for APIC. To coexist with 8259, the APIC interrupt input connected to 8259 will be programmed for virtual wire mode. In virtual wire mode, the time window of 8259 will apply. So the driver will behave same way as it was behaving with 8259.

Consideration 4

Question: EISA system has active low level polarity. 82489DX itself does not have polarity control register to support this EISA feature. Implementations using external polarity register may implement the polarity register at different address. Will this introduce a problem for achieving the goal of single binary?

Response: 8259 by itself does not have polarity control whereas 8259 when implemented in EISA chipset has the polarity control. Similarly APIC does not have by itself polarity register. When implemented in ESC chipset, it will have polarity control register. So polarity register should be programmed as a part of EISA BIOS and not APIC BIOS. Since system BIOS or EISA BIOS should be able to take charge of changes, if any, to polarity control register. APIC BIOS should not be affected by differences in the address for polarity register.

Consideration 5

Question: 8259 recognizes the interrupt when an edge occurs at the interrupt input pin even if the interrupt was masked. So when the interrupt input is later unmasked, the interrupt is posted to the CPU. 82489DX does not register this edge and if interrupt happens when the interrupt is masked 82489DX just ignores the interrupt. When later it is unmasked there is no further edge and so 82489DX never passes that forgotten edge and that interrupt channel becomes unusable after that.

Response: When the interrupt is masked, logically interrupt controller should ignore whatever happens there. It is strongly recommended that first 82489DX should be unmasked and then the device interrupt should be enabled. By this sequence, software can ensure that always an edge will occur at the APIC input only after the interrupt is unmasked.

Please contact Intel for platform level specification in Multiprocessor system design using APIC.

APPLICATION NOTE

3

82489DX User's Manual

M. JAYAKUMAR
MULTIPROCESSING TECHNOLOGY GROUP

October 1993

PRELIMINARY

3-1

82489DX User's Manual

CONTENTS

	PAGE
INTRODUCTION	3-3
REGISTER ORGANIZATION	3-3
INITIAL REGISTER VALUES AFTER HARDWARE RESET	3-3
SYSTEM CONSIDERATIONS WHILE PROGRAMMING THE 82489DX	3-3
82489DX and Memory Mapping	3-3
Unique ID Requirement	3-3
PROGRAMMING THE LOCAL UNIT	3-4
Do's and Don'ts	3-4
Atomic Write Read to Task Priority Register	3-4
Task Priority Register and Total Usable Vectors	3-4
ISR/IRR/TMR	3-4
Interrupt Command Register Programming Considerations	3-4
Critical Regions and Mutual Exclusion	3-5
Buffering in Interrupt Command Register	3-5
Interrupt Command Register DOs and Don'ts	3-5
IPI through Interrupt Command Register	3-6
ExtINTA Interrupt Posting	3-6
Lowest Priority	3-6
Disabling Local Unit	3-7
Issuing EOI	3-7
External Interrupts and EOI	3-7
Spurious Interrupts and EOI	3-7
NMI and EOI	3-7
PROGRAMMING I/O UNIT	3-7
Interrupt Sharing Considerations	3-7
I/O Unit and Priority	3-7
MP SYSTEM	3-7
Initialization Sequence	3-7

CONTENTS

	PAGE
Section A	3-8
Write the Local Unit ID (if needed)	3-8
Write All Ones to Destination Format Register	3-8
Write to Logical Destination Register	3-8
Raise the Task Priority	3-9
Program the Spurious Interrupt Vector and Enable the Local Unit	3-9
Program the Vectors for Local Interrupts and Timer	3-9
Program the Timer Control Registers	3-9
Clear the Interrupt Mask for Timer and Local Interrupts	3-9
Initialize the Local Interrupt Sources ...	3-9
Broadcast ALL.INCL.SELF Reset Deassert Message	3-9
Lower the Task Priority	3-9
Section B	3-9
Synchronization	3-9
Section C	3-10
System Wide Resources Programming	3-10
INTERRUPT SERVICE ROUTINE	3-10
ISR(x)	3-10
DOS Environment	3-11
Transition from 8259 to 82489DX	3-11
Spurious Interrupt Service Routine	3-12
Spl(x) Routine	3-12
82489DX AND PCI-EISA BRIDGE INTEROPERABILITY	3-14
HARDWARE DESIGN CONSIDERATIONS	3-14
REGISTER PROGRAMMING DETAILS	3-16
CONCLUSION	3-26

INTRODUCTION

82489DX is the new interrupt controller for high performance systems and 32-bit OS. Some important considerations for hardware designers are given. This application note will provide information of all registers in 82489DX and their bits and bytes organization. The control word for various programming options are given in a tabular format. Some programming hints are given to facilitate a quick understanding of the interrupt architecture and the priority model in 82489DX.

The programming model discusses the registers, their data structure like fields, bits, bytes and default register values. The system considerations and key points to be noted while programming 82489DX are discussed next. Typical examples of initialization, interrupt service routine and Spl() routines are given. The notes discuss important hardware design considerations.

Related Reference Materials

- 1) 82489DX Data Book, Order Number 290446.
- 2) An APIC based Symmetric Multiprocessor System Design AP-474, Order Number 241521.

REGISTER ORGANIZATION

The 82489DX contains both the local unit and I/O unit. I/O unit has its own Unit ID and local unit has its own Unit ID. Both units are operational at all times once they are enabled and the access can be done to both units. It should be noted that the local unit has its own version register, and I/O unit has its own version register, namely, I/O version register. The unit enable bit is provided for local unit and it is not provided for I/O unit. However, I/O unit has mask bit for each redirection table entry to mask the interrupts. Functionally I/O unit can only transmit interrupt messages whereas local unit can both transmit and receive interrupt messages. In summary, 82489DX should be viewed as an integrated chip having a local unit and an I/O unit both capable of operating at the same time.

INITIAL REGISTER VALUES AFTER HARDWARE RESET

The local unit ID register latches the value on the address pins A3 to A10 after hardware reset whereas the I/O unit ID register gets cleared to 0 after hardware reset. The local unit Version Register is cleared to 0 whereas the I/O unit Version Register contains 1111 in

its Max Redir Entry field. The interrupt masks in the local timer vector table register and in the I/O redirection table entry(31:0) registers are set so that after reset all the interrupts are masked. The spurious vector register's unit enable bit is cleared so that local unit is disabled after hardware reset. Since all the interrupts are masked after hardware reset, the I/O unit will not transmit any interrupt after hardware reset until mask is cleared specifically by software and the interrupt is active.

All other registers are cleared to 0 after hardware reset.

SYSTEM CONSIDERATIONS WHILE PROGRAMMING THE 82489DX

The 82489DX register data structure contains different fields to specify the mode of operations and the options available within each mode. Since certain options are applicable to specific modes only (for example "Remote Read" mode applies only to Interrupt Command Register, it does not have any relevance to I/O unit's redirection tables) the following programming hints are provided.

82489DX and Memory Mapping

The 82489DX is a 32-bit high performance interrupt controller. It allows the CPU to do 32-bit read and write to it. By memory mapping the 82489DX, system performance can be enhanced. Even though the 82489DX can be memory mapped, its functionality as an interrupt controller should be kept in mind while programming the virtual memory management control data structure. The caching policy for the page where an 82489DX is mapped should also be done with the functionality of the 82489DX in mind. For example, the reads to an 82489DX should not be cached and writes should be write-through. Since 82489DX registers are aligned at 128-bit boundaries, memory mapping the 82489DX with interleaved memory system should not be a problem. However, it should be noted that the 82489DX does not support pipelining.

Unique ID Requirement

All the local units and I/O units hooked on an I_{CC} bus should have a unique ID before they can use the bus. This should be ensured by the programmer, since for I_{CC} bus arbitration the units (whether it is local unit or I/O unit) arbitrate with their unit ID.

PROGRAMMING THE LOCAL UNIT

Dos and Don'ts

1. The local interrupt vector table entry (and the I/O unit redirection table entry) should not be programmed for "Remote Read" Delivery mode. In other words, only Interrupt Command register supports "Remote Read" Delivery mode.
2. Local Interrupts should not be programmed with "Lowest Priority" Delivery mode.
3. Local Interrupts should not be programmed with "Reset" Delivery mode.
4. It is not recommended to use level triggered mode except for "Reset Deassert" messages.

Atomic Write Read to Task Priority Register

This section discusses issues regarding write buffer flushing and necessity of atomicity of task priority register programming.

Typically, the task priority register is written with higher priority to mask certain low level interrupts before entering into a critical section code. In a system where an 82489DX is memory mapped the CPU may buffer this task priority register write to its on chip write buffer. The following scenario can happen in such situation: CPU posts task priority register write to its on chip write buffer and enters into the critical code. A lower priority interrupt (which should not enter the critical code) interrupts the CPU before the write buffer gets flushed into task priority register. The CPU now erroneously accepts the lower priority interrupt. To avoid the situation, atomic write and read to task priority register should be done. The read following write ensures that the write buffer is flushed to task priority register and the atomicity ensures that no interrupt will be accepted by the CPU during its write to task priority. In case if the CPU itself takes care of flushing its write buffers before INTA cycle, there is no problem. However, if there are system posted write buffers then external logic should make sure to flush the system write buffers before INTA-cycle.

Task Priority Register and Total Usable Vectors

Task priority register is used to specify the priority of the task the processor is executing. In 8259 the priority is defined only among the interrupts that it handles. 82489DX goes further ahead in handling priority. In multitasking system, in addition to device interrupts, various tasks have different priority and 82489DX allows consideration of the priority at system level. The processor specifies the priority of the task it executes by

writing to task priority register. Now any interrupts *at or below* the task priority will be masked until the task priority gets lowered. The masking granularity is at priority level. Out of 256 interrupt vectors 16 priority levels are specified and 16 vectors share one priority level. Since the masking granularity by the task priority register is at priority level, group of 16 vectors get masked when a local unit increases its task priority by one level.

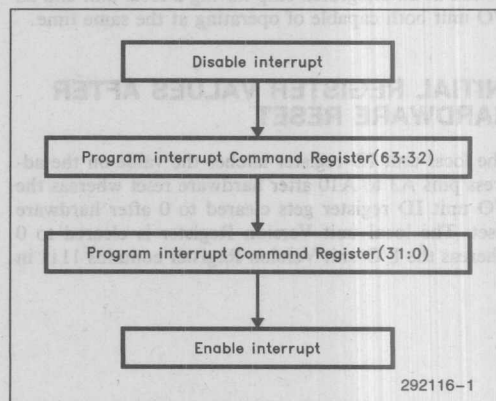
When task priority register is at its minimum level of 0, interrupt vectors having level 1 to 16 are passed to CPU. Stated in other words, even when the task priority register is at its minimum (level of 0), interrupt vectors at level 0 will be masked. This means that the interrupts should not be programmed with vectors 0 to 15. So out of 256 interrupt vectors, only 240 interrupt vectors (vector 16 to 255) can be used in 82489DX.

ISR/IRR/TMR

1. Bits 0-15 of IRR/ISR/TMR do not track interrupt. No interrupt of vector number from 0-15 can be posted. The total interrupts supported are 240. This can be easily explained by the way the priority mechanism is defined. When reading the lowest 32 bits of this register, 0 will always be returned for the lower 16 bits.

Interrupt Command Register Programming Considerations

The interrupt command register (31:0) has the side effect of sending interrupt once it is written. There is no mask bit associated with Interrupt Command Register. Once interrupt command register (31:0) is written, the interrupt is sent from the local unit. The interrupt destination is provided in the interrupt command register (63:32). So, the interrupt command register (63:32) should always be programmed before the interrupt command register (31:0) is programmed.



CRITICAL REGIONS AND MUTUAL EXCLUSION

This section discusses the reasons for mutual exclusion to be exercised when writing to interrupt command register. Each 82489DX has a single Interrupt Command Register that is used to send interrupts to other processors. The programmer should make sure to synchronize access to this register. Specifically, (1) writing all fields of the register (MSB), (2) Sending the interrupt message (by writing the LSB register), and (3) waiting for Delivery State to become Idle again, should occur as a single atomic operation. For example, if interrupt handlers are also allowed to send inter processor interrupts, then interrupt dispensing to the processor must be disabled for the duration of these activities so that interrupt handlers are excluded from accessing the ICR. This is explained as follows. Let us assume in a typical MP system preemptive scheduling (on another processor) is implemented by sending inter processor interrupts (IPIs). IPI can be also used for clock distribution in an asymmetric system where the timer interrupts only one processor and that processor notifies all other processors in the system through IPI. Inter processor interrupts are implemented by using interrupt command register. If we allow interrupts during writing interrupt command register the following erroneous operation may result. If interrupts are enabled (they should not be) during writing to interrupt command register, interrupts can come after writing to MSB portion and before writing to LSB portion of interrupt command register. Now in the interrupt service routine, if ICR is used (for distribution of interrupt to other processor(s), for example) then this ISR also starts writing to the Interrupt Command Register. That means the ISR will overwrite the MSB portion just written by the previous IPI. After returning from the ISR when the previous IPI continuous writing to the remaining LSB portion, the message will be delivered to wrong address since MSB is modified by the module which interrupted. **The inference is that while accessing ICR interrupts should be disabled.** Also it should be noted that except for "Reassert Deassert Messages", IPI should only use edge triggered mode.

BUFFERING IN INTERRUPT COMMAND REGISTER

The Interrupt Command Register provides one level of buffering which should be kept in mind while programming an 82489DX. The ICR (Interrupt Command Register) becomes busy as soon as inter processor message is written into it. It hands the message over to ICC bus transmit unit which in turn tries to send through ICC bus. Since the ICR has passed the command to transmit Unit (whose responsibility is to send it through ICC bus) it becomes free. The software before writing next inter processor message reads the flag to be free and writes next message. Thus there is a possibility of next message being written into the 82489DX before the first message is really sent out. The programmer should be aware of this.

INTERRUPT COMMAND REGISTER DOS AND DON'TS

1. "ExtINTA" delivery mode should not be used for all destination shorthand.
2. "Remote Read" should always be programmed as "Edge" triggered interrupt.
3. "Remote Read" should always be programmed with physical Destination mode (and not with Logical Destination mode).
3. Only Fixed Delivery Mode should be used for "Self" destination shorthand. Stated otherwise, "lowest priority", "Remote Read", "Reset", "NMI" delivery modes do not apply for "Self".
4. For "All incl. self" and "All excl Self" destination shorthands, "Remote Read" delivery mode should not be used.
5. For "All incl. self" and "Self" destination shorthands "Reset" Assert mode should not be used.
6. For "All exclusive self" destination shorthand if "Reset ASSERT" delivery mode is used, it should be ensured at system level that only one processor executes this instruction at any time. To explain this, let us consider the following situation. Let us assume that two CPUs, CPU A and CPU B are executing "Reset ASSERT, All Exclusive Self". The message of CPU A puts every CPU except CPU A in reset state. After the message is written by CPU A it typically takes 2.9 μ s for the message to flow through the ICC bus to reach other local units to reset all other processors. Before this message resets, let us assume another processor also, say CPU B, issues the "Reset ASSERT, All Exclusive Self" message. The following CPU B message (which was sent out before CPU B itself got reset because of CPU A reset message) will reset every CPU, which will include CPU A, except CPU B. But CPU B will eventually get reset by the message sent by CPU A and CPU A will also get reset by the message sent by CPU B. Thus all the CPUs in the system goes into reset state and this is an irrecoverable state. To avoid this, only one processor should execute this instruction at any time. This can be achieved, for example, by spinlock or mutex implemented as shared variables between multiprocessors.
7. Messages could be sent out in "Logical" or "Physical" mode with destination ID of all 1's depending on the way Destination Mode entry is programmed. In brief, "All incl. self" and "All excl. self" supports both "Logical" and "Physical" addressing mode.
8. When destination shorthand (i.e., broadcast) is used with "lowest priority" destination mode, then even though all participates in arbitrating for destination, only the lowest priority gets the message. So even though the addressing is broadcast since the destination mode is lowest priority only one gets the message.

with "Fixed" destination mode, then all the units get the message. So to send messages to all units Fixed destination mode should be used in addition to using destination shorthand.

10. It is recommended that all IPI messages, except for "Reset Deassert", use only edge triggered interrupt mode.

IPI THROUGH INTERRUPT COMMAND REGISTER

Interrupt command register can be used to send inter processor interrupt. Inter processor interrupts can be used for preemptive scheduling, TLB flushing, clock distribution, etc. IPIs can also be used in asymmetric systems to pass certain work to another processor who has exclusive access to certain piece of hardware. Let us consider a dual processor system which uses only two 82489DX in the whole system. Since the local units should be accessible only by their respective processor a local unit should be selected only when the arbitrator grants the bus to its processor. Since 82489DX has common chip select for its local unit and I/O unit, for logical simplicity, system hardware may select a 82489DX when the corresponding processor is granted bus. Because of this, processor A can access only I/O unit and local unit that are available in its 82489DX. It is not possible to access the I/O unit of the other 82489DX. The same thing holds good for the other processor. Since I/O unit should be globally visible to both processors, there may be situations when a processor may want to access the other I/O unit. This is typically the case for enabling and disabling the I/O interrupt. IPI can be used to pass that task to the other processor which can access that I/O unit. This is just one example for using IPI.

ExtINTA INTERRUPT POSTING

ExtINTA interrupts are used to support 8259 in a 82489DX based system. The external interrupts (ExtINTA) are specific in their characteristics in that they do not have any priority relationship with rest of the interrupt structure. But when posting an interrupt to the processor, if both an external interrupt and a 82489DX interrupt are pending, 82489DX could post either one to the processor. In 82489DX implementation, it would post external interrupt whenever there is no other 82489DX interrupt that can be posted to the processor. It should be also noted that External Interrupts can not be masked by raising task priority. However, they can be masked by the mask bit in the table entry for that (ExtINTA) interrupt.

Since ExtINTA interrupts do not have any priority relationship, ISR and IRR bits are not maintained for

concerned, if more than one ExtINTA interrupts are directed towards a local unit, that local unit treats all the ExtINTA interrupts directed to it as only one ExtINTA interrupt. This leads to an important point that in a system no more than one interrupt should be programmed as ExtINTA interrupt type with the same destination. However, it should be noted that there can be more than one ExtINTA type of interrupt in a system with each having different local unit as destination.

LOWEST PRIORITY

Under the lowest priority delivery method, the processor to handle the interrupt is the one in the specified destination with the lowest processor priority value. If more than one processor is at the lowest priority, then a unique arbitration ID is used to break ties. To have unique arbitration ID in the system (which is mandatory for the lowest priority algorithm to work) all the arbitration ID of local 82489DXs in the system should be in sync. On reset, arbitration ID is reset to zero by the hardware. Hence all the local units in the system after reset will have same arbitration ID (namely zero). For lowest priority arbitration to work properly we need to have unique arbitration ID in the system. This means after local unit IDs are written in all local units (obviously, each unit ID should be different from other IDs) a RESET DEASSERT message should be sent in ALL INCLUSIVE mode. The important side effect of RESET DEASSERT message is that it copies the unitID into the respective arbitration ID. Since unit IDs are unique, the RESET DEASSERT message ensures that the arbitration ID also are unique in the system. This RESET DEASSERT message should be sent before system is used for lowest priority arbitration.

The RESET DEASSERT message, if not sent, only once delivery semantics may not be guaranteed. If RESET DEASSERT message is not sent then all the arbID in the system will be same. When a message is sent in the lowest priority arbitration, the participating local units use their processor priority concatenated with arbitration ID to decide the destination. Processor priority is derived from the task priority. There is a chance that two local units can have same task priority depending on the code they are executing and thereby same processor priority. In addition since arbID are also same if RESET DEASSERT message WAS NOT sent, all the processors in the same priority may accept the message in lowest priority arbitration. This violates the only once delivery semantics. The inference is that RESET DEASSERT message in ALL INCLUSIVE SELF mode should be sent as part of initialization before enabling interrupt in the lowest priority destination scheme.

It should be noted that only once delivery semantics for a group destination is guaranteed only if multiple fixed delivery of the same interrupt vector are not mixed.

DISABLING LOCAL UNIT

Once the 82489DX is enabled by setting bit 8 of spurious vector register to 1, the user should not disable the local unit by resetting the bit to 0. The result will put the local unit in an inconsistent state. However, a local unit can be disabled by getting "reset" interrupt message from any other local unit across the ICC bus.

ISSUING EOI

EOI, End of Interrupt issuing indicates end of service routine to 82489DX. Always the highest priority ISR bit which is set during INTA cycle gets cleared by EOI. This section discusses the relevance of EOI to the specific types of interrupts and its timing related to interrupt deassertion.

EXTERNAL INTERRUPTS AND EOI

External Interrupts (ExtINTA) should be programmed as edge type. INTA cycles to external interrupts are taken automatically as EOI by 82489DX. This is similar to AEOI, Automatic End of Interrupt of 8259A. So EOI should not be issued to 82489DX for ExtINTA interrupt servicing. For ExtINTA type of interrupts, there is no need to have interrupt service routines since the main purpose of ExtINTA interrupt itself is to have software transparency in the compatible mode. The existing interrupt service routines written for 8259 will be executed by the processor for ExtINTA interrupts.

SPURIOUS INTERRUPTS AND EOI

Spurious Interrupts do not have any priority relationship to other interrupts in the system. So IRR is not set for spurious interrupts. EOI should not be issued for spurious interrupts. It is advisable not to share the spurious interrupt vector with any interrupt.

If spurious interrupt vector is shared with some other interrupt then the following guidelines should be followed. If the source is spurious interrupt (for which the corresponding ISR is not set) then EOI should not be issued. If the source is a valid interrupt sharing the spurious interrupt vector (for which the corresponding ISR is set) then EOI should be issued.

NMI AND EOI

For NMI type of interrupt no IRR bit is set. So, obviously EOI should not be issued while servicing NMI type of interrupts.

PROGRAMMING I/O UNIT

Interrupt Sharing Considerations

Two different interrupts should not be programmed with the same interrupt vector. This means that each redirection table in a system should have unique vector. Interrupt sharing can be done electrically. Interrupts connected at different interrupt input pins of 82489DX CAN NOT share interrupt by having same vector. 82489DX does not support active low interrupts. So sharing interrupts should have polarity logic support externally.

I/O Unit and Priority

The 82489DX partitions its interrupt control function among two different units:

1. I/O unit
2. local unit

The priority resolving is done at local unit. The I/O unit does not involve itself in the priority mechanism. The I/O unit takes a snapshot of interrupts pending at the INTIN interrupt input pins. If interrupts are active, it starts sending the interrupt messages over ICC bus. It starts sending the lowest numbered interrupt input first. That is if INTIN0 and INTIN5 are found active in a snapshot, interrupt message corresponding to INTIN0 is sent first regardless of the priority of the vectors that are associated with these interrupts. It sequentially sends all the interrupts found active in a snapshot. Before sending, it checks whether the corresponding INTIN is still active. **This is the reason why interrupts, both edge and level triggered, should be kept active until CPU acknowledges it.** The difference between edge triggered and level triggered interrupt is that edge triggered interrupts ensure only one activation of interrupt per low to high edge whereas the level triggered interrupt allows to have multiple interrupts as long as the interrupt is held high. It should be noted that both edge and level triggered interrupts are active high.

MP SYSTEM

Initialization Sequence

This section assumes the system with multiple CPUs with each CPU having its own 82489DX local units and local interrupts (like local secondary cache data parity interrupt, coprocessor interrupt etc.,) connected to the respective local units. The system additionally assumes symmetric multiprocessing in the sense that I/O system is symmetric and it can be initialized by any CPU in the system.

Section A: Code Executed by all CPUs in the system

Section B: Synchronization to indicate Section A is completed

Section C: Only one CPU need to execute this Code

Each local unit is visible (through address mapping) only to that CPU to which local unit is attached. So each local unit will be programmed by its own CPU. Thus the code specified as section A will be executed by all CPUs in the system.

Section B of the initialization code is also executed by all the CPUs. This section of the code ensures that all the CPUs have completed execution of their "Section A" so that all Local units are properly initialized with different IDs, the system is in a consistent state, etc.,

Section C initializes system wide I/O unit and enables the interrupt mechanism to start functioning. Since the I/O unit is system wide, only one CPU need to program the I/O unit part of the 82489DX.

Section A

Write the local Unit ID (if needed)

Write all Ones to Destination Format Register

Write Logical Destination Register

Raise the Task Priority

Program the Spurious Interrupt Vector
Vector and Enable the Local Unit

Program the Vectors for Local Interrupts and
Timer

Program the Timer Control Registers

Clear the mask for Local Interrupts and Timer

Initialize the local Interrupt sources

Broadcast ALL INCL. SELF
Reset DEASSERT message

Lower the Task Priority

It should be noted that the interrupt descriptors, interrupt service routine, spurious interrupt service routine and other interrupt related structure should have been initialized before the Section A. This is because Section A code enables respective local interrupts and timer interrupt vectors and when the interrupts arrive from these devices Section A ensures that 82489DX will provide the vector. But the code executed before Section A should ensure the interrupt structure is initialized. Spurious interrupts are bound to occur because of the asynchronous interaction between interrupts and software writing to task priority register. So spurious interrupt service routine has to be initialized before Section A.

WRITE THE LOCAL UNIT ID (IF NEEDED)

Each local unit can get the ID latched by reset from 82489DX address pins A3-A10. If the hardware ensures that during reset each local unit in the system gets different pattern on the address pins A3-A10 then all the local units are initialized automatically with different IDs. In that case writing to the local unit ID by software is not mandatory. If software writes the local unit ID then it should be read from some address space which is same for all CPUs but have different IDs for different CPUs. This will ensure that the same code when executed by different CPUs will initialize respective local unit with different ID.

WRITE ALL ONES TO DESTINATION FORMAT REGISTER

All the 32 bits of Destination Format Register are written with 1. This is to support single level logical addressing mode. This mode is explained in the following paragraph.

WRITE TO LOGICAL DESTINATION REGISTER

The logical Destination Register should be written with the logical destination address. It should be noted that since each CPU needs to assign a different logical Destination address to its own 82489DX local unit and since this code is executed by all CPUs the logical Destination address should be read from some address space which is the same for all CPUs but contains different Destination address values. Since logical Destination address is in bit decoding format, typically this can be achieved by shifting the CPUID.

PRELIMINARY

The logical destination register with Destination format register can be used to support flat model. In this model, bits 24 through 31 of the destination address of the interrupt message vector are interpreted as decoded field. Intel strongly recommends for future compatibility to use only bits 31 to 24 of logical destination register. To have binary compatibility with future APIC implementations, any code written for 82489DX should not use bits 0 to 23 of logical destination register. This field is compared against the logical destination register of the local unit. If there is a bit match (i.e., if at least one of the corresponding pair of bits of the destination field and logical destination register match) this local unit is selected for interrupt delivery. Each bit position in the destination field corresponds to an individual local unit. For future compatibility, only bits 0 to 23 of logical destination register should be zero. This scheme allows the specification of arbitrary groups of 82489DXs simply by setting the member's bit to one, but allows a maximum of 8 local units in the system since bits 0 to 23 of the logical destination register is zero. Broadcast to all is achieved by setting all 8 bits of destination to ones. This selects all 82489DXs in the system.

If more than 8 units are to be addressed in the system (and if future compatibility is not a major issue) then all the bits of the logical destination register can be used as a bit map thereby increasing the number of CPUs addressable in logical addressing to 32.

RAISE THE TASK PRIORITY

Before enabling the local interrupts and timer interrupts the task priority is raised to maximum priority in the system so that these interrupts are masked temporarily.

PROGRAM THE SPURIOUS INTERRUPT VECTOR AND ENABLE THE LOCAL UNIT

The spurious interrupt vector register is programmed with the corresponding vector. This vector will be pointing to a dummy routine with just an IRET. The unit is enabled so that the tristate pin PINT can come out of tristate state to pass the interrupts.

PROGRAM THE VECTORS FOR LOCAL INTERRUPTS AND TIMER

The interrupt vectors for Local Interrupts and timer are initialized with corresponding vector.

PROGRAM THE TIMER CONTROL REGISTERS

The timer registers such as divider configuration register, initial count, mode of operation and source of the timer clock are programmed.

CLEAR THE INTERRUPT MASK FOR TIMER AND LOCAL INTERRUPT

The interrupt mask is cleared for timer and local interrupts by clearing the interrupt mask bit in their respective interrupt vector register

INITIALIZE THE LOCAL INTERRUPT SOURCE

The local interrupt sources are also programmed for proper system operation. This involves enabling the interrupt from the sources. The order of enabling the interrupt is very important. First the 82489DX entries should be cleared and then the sources connected to the pins should be enabled. If done the other way, interrupts may get lost. This is true particularly in edge triggered interrupt inputs where if 82489DX mask is cleared after enabling the source interrupt, 82489DX may not have a chance to capture the low to high edge which might have produced immediately after the source interrupt is enabled and before 82489DX mask is cleared.

BROADCAST ALL INCL. SELF RESET DEASSERT MESSAGE

This is done so that all the local unit's ArbIDs are in sync. It should be noted that for breaking the tie during lowest priority arbitration ArbID is used. ArbID is copied from local unit ID during reset. Since local unit IDs can be written through software and at that time ArbID is not updated there may be a case where all ArbID in a system to have same value. To avoid such situation Reset Deassert message is sent to ALL INCL. SELF so that the ArbIDs are different in the system.

LOWER THE TASK PRIORITY

Task priority is lowered so that the interrupts can be armed to the CPU.

Section B

Synchronization

There are many methods available for synchronization. **Test - and - set** is a simple primitive, for example, available for synchronization. **Counting semaphores** can be built using this test - and - set primitive and synchronization can be achieved.

The main idea is to achieve global synchronization among the processors to indicate the local unit portion is programmed.

Section C

SYSTEM WIDE RESOURCES PROGRAMMING

This portion needs to be programmed by one CPU only. It should be noted that since the system environment we are assuming is shared memory symmetric MP system, CPU specific coding is not possible. System wide resource programming can be achieved by many ways depending on simplicity and performance (since this is only initialization routines, performance should not matter much) tradeoff. The following sequence illustrates a simple approach to program. The assumption here is that 82489DX will get reset both during cold reset and warm reset.

Locked access to the system wide resource,
I/O unit

Read a **specific MASK** from
Redirection Table Entry

If the mask is set, Jump to **Prog. I/O unit**

If the mask is not set, **Release lock** and Jump
to **I/O unit Done**

Prog. I/O Unit: Write to the index register to
select unit ID reg

Write I/O unit ID in the ID register

Write to index register to select I/O unit
Version Reg

Read the Version reg. to know no. of
RedirTable Entries, **N**

RedirTble: Write to index register to address
MSB Redir. Table **n**

Write to MSB Redirection Table Entry **n**
Destination local unit ID

Write to index register to address LSB
Redir. Table **n**

Write to LSB Redirection Table **n**
mode,dest.,mask, Vector of INT

Loop to RedirTble: till all N (here N = 16)
Entries are done

Release the lock

I/O unit done: Remaining system init like I/O
system etc.,

The first CPU getting the lock will find the mask to be set (since after reset, 82489DX mask is set). It locks the I/O unit for programming. The mask selected is specific in that the system initializes such that the mask is cleared during initialization. So by reading that mask mutual exclusion is achieved. If the system requirement is such that no mask can be cleared during system initialization some other register can be read. For example, the I/O unit IDs are reset to 0 on reset. Since the system initialization will have all the local unit IDs starting from 0 and I/O unit will be initialized by the system to non Zero ID, I/O unit can be read and if 0 can be assumed that programming is not yet done (so that it can gain control of lock and start programming) and if found non Zero, then that CPU can skip programming the I/O units by jumping to I/O unit done.

The I/O unit registers are organized as index register and data register. Other portions of section C are self explanatory. After I/O unit done, the I/O system initialization can be started so that the interrupts can start flowing in the system.

INTERRUPT SERVICE ROUTINE

ISR (x)

Save Stack and frame pointer for
parameter referring

Save hardware context and software context

Service: Service the source, modify shared data
structure etc.,

EOI: Issue EOI to reset ISR of level x in
82489DX

Restore hardware context and software context

Restore Stack and Return from Interrupt

The above ISR x shows the interrupt service routine of interrupt level x. The stack, hardware context like CPU registers and software context like task specific variables are saved. The Servicing is done as specific to the interrupting source. This may involve reading a status register or initiating a thread to read a "full" buffer, initiating a thread to write data to some "empty" register, or acknowledging an interrupt from another CPU. This is the point at which the interrupting source is supposed to deactivate its request. Next EOI is issued to reset the ISR bit corresponding to the interrupt level x. Till now the interrupts from same level and lower levels were masked. Once EOI is written, interrupts from all the levels can start coming. The hardware context and software context are restored followed by stack cleaning and a proper Return from Interrupt is executed.

There are couple of timing issues that can be considered here. The time delay between Service and EOI is referred here. This timing and its relevance to edge/level triggered interrupt is discussed as follows: In the case of edge triggered interrupts, for each edge one Interrupt message is sent by I/O unit to local unit over I_{CC} bus whereas for level triggered interrupts there are two interrupt messages sent, one during assertion of level interrupt, and another during deassertion of level interrupt. In edge triggered interrupts, since the deassertion of interrupt does not result in any interrupt message, there are not many issues with the timing delay between Service and EOI, even though in general delaying EOI means interrupts from the same interrupt source are kept pending from interrupting CPU. In level triggered interrupts after service the I/O device starts deasserting its interrupt request. This results in an interrupt message to clear IRR bit in the local unit. This may take some time because the minimum possible time in I_{CC} bus is 2.3 μ s (10 MHz I_{CC} clock assumed). If the I_{CC} bus is occupied by some other messages already then this IRR clearing message has to wait to get its turn which means additional delay. If EOI is issued before this happens then ISR gets cleared and IRR for this "done interrupt" is still alive to erroneously set ISR again. This will result in another interrupt. So "Early Servicing" is advisable in level triggered interrupts.

DOS Environment

In the DOS environment the initialization portion is the only routine to be coded since the 82489DX acts as a virtual wire once initialized and needs no more programming. Since it is uniprocessor environment there is no need for synchronization.

The interrupt from 8259 is programmed as type ExtINTA and other redirection table entries are not accessed since their masks are set by reset and hence disabled.

In the Interrupt service routine, since EOI is not needed for ExtINTA type of interrupts, no programming is needed for 82489DX. Since ExtINTA type of interrupts do not have any relationship to task priority, Spl routines do not apply for DOS configurations.

Transition from 8259 to 82489DX

Typically, platforms with 82489DX will support 82489DX in virtual wire mode. The BIOS in the EPROM will program the 82489DX in "Virtual Wire" mode. Typically systems boot DOS and then the 32 bit high performance OS is given control. There are also situations where after BIOS code is executed the high performance OS is given control. In both the situations, the 8259 will be operational during the DOS or BIOS portion of the code and interrupts will be flowing in the system. When the high performance OS is given control, it may want to disable the interrupt during initialization. This will involve disabling 8259. After disabling 8259, the 32 bit OS initializes and then it may want to enable interrupt mechanism which involves enabling 82489DX. The sequence we are encountering here is 8259 (and one input of 82489DX enabled in "ExtINTA" mode) enabled, 8259 disabled and then 82489DX enabled. When 82489DX is enabled in 32 bit OS all the interrupt inputs are enabled as opposed to the only one interrupt enabled in "Virtual Wire" mode. The additional difference is that 82489DX is no more a "virtual wire" but it is functioning as an interrupt controller.

In the above situation, consider the following scenario. The 82489DX is functioning as "virtual wire" and passing the 8259 interrupts as "ExtINTA" mode to the local unit. When interrupt mechanism is disabled by CLI (Clear interrupt) or masking the 8259 interrupt, there may be a possibility that already 8259 originated interrupt may be pending at the local unit asserting interrupt to the CPU. Now since the CPU has executed CLI, the interrupt is not serviced and the interrupt is kept pending. It should be noted that the pending interrupt is of type "ExtINTA". After this, 32 bit OS gets loaded which configures 82489DX redirection tables and interrupt is enabled. Now the "old pending" interrupt is delivered and since it is "ExtINTA" the external hardware will typically pass the interrupt acknowledge cycle to 8259. But at this point of time 8259 has been masked by 32 bit OS. Hence the "masked" 8259 responds with IR7 vector. So the 32 bit OS should reserve IR7 vector for both master and slave 8259 for "emptying" the old pending interrupt since the "Virtual Wire" remembers the previous interrupt.

Sequence of Enabling: In the case of enabling interrupt controllers in "ExtINTA" mode the 82489DX should be enabled before the 8259 interrupt Controller is enabled. This is because ExtINTA is "edge triggered" and if 8259 is enabled before 82489DX, 8259 might have

given an interrupt request by activating its interrupt output while 82489DX is still not enabled. When 82489DX is enabled the interrupt input has a high level and 82489DX had no chance of capturing the low to high edge of 8259.

Spurious Interrupt Service Routine

It is advisable not to share spurious interrupt vector with any genuine interrupt source. This section assumes that spurious interrupt vector is not shared with any other interrupt.

82489DX does not set ISR in response to spurious interrupt, NMI type of interrupt, Reset type of interrupt and ExtINTA type of interrupts. For all these interrupts EOI should not be issued.

Some systems have a variable count in the supervisor data structure to count number of spurious interrupts raised in the system. This can be used to study the reliability and "noise level" of the system. But in 82489DX architecture, spurious interrupt can occur even by a dynamic write to task priority register, frequency of spurious interrupt does not mean anything related to "noise level".

Return from interrupt

Spl(x) Routines

The processor handles the I/O system through device driver interface. The device driver consists of two entries to access the I/O system: 1) Call entry and 2)

Typical usage of these routines

```
y = spl() //Save the current task priority register value//
spl(x) //Raise the task priority value //
:
:
: // Access the shared data structure //
spl(y) // Restore the task priority register //
```

Interrupt entry. The interrupt entry is the one that we have been discussing for a while, i.e., interrupt service routine. The Call entry is the way the I/O system is accessed to initiate and service devices. The call entry has its own task priority and interrupt entry has the priority that is associated with the device interrupt level. The call entry and interrupt entry processes have I/O data structure like linked list, buffer pointers in common which they share. Mutual exclusion is needed to ensure the integrity of I/O system.

To ensure the mutual exclusion between these two processes running in the same processor, Spl(x) routine is used. The call entry routine (which is normally at a lower priority than the interrupt entry routine) calls Spl(x) routine to elevate its own priority above (or equal to) that of the corresponding device's interrupt priority. At this priority the interrupts from the device are masked out and the shared data structures can be accessed (exclusively).

Once this is done the priority is restored back to original value so that other interrupts won't suffer for relatively long time.

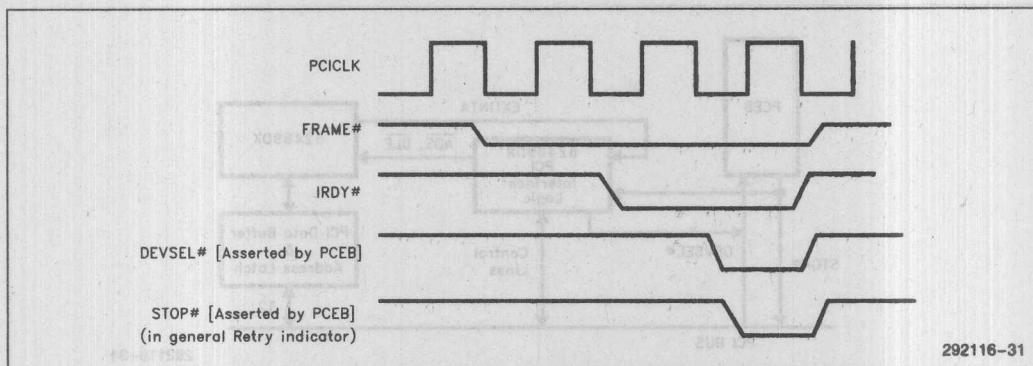
Spl() is used to save the current task priority and Spl(x) is used to elevate the task priority.

Spl()

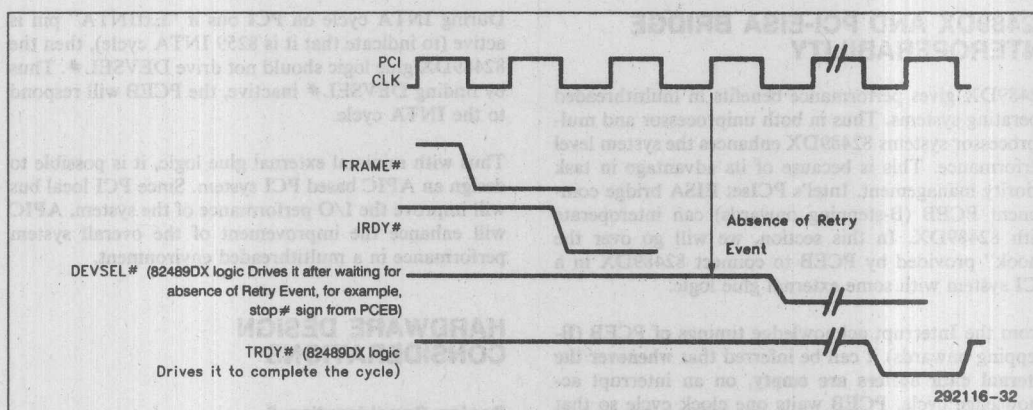
Read and return the 82489DX
local unit task priority register

Spl(x)

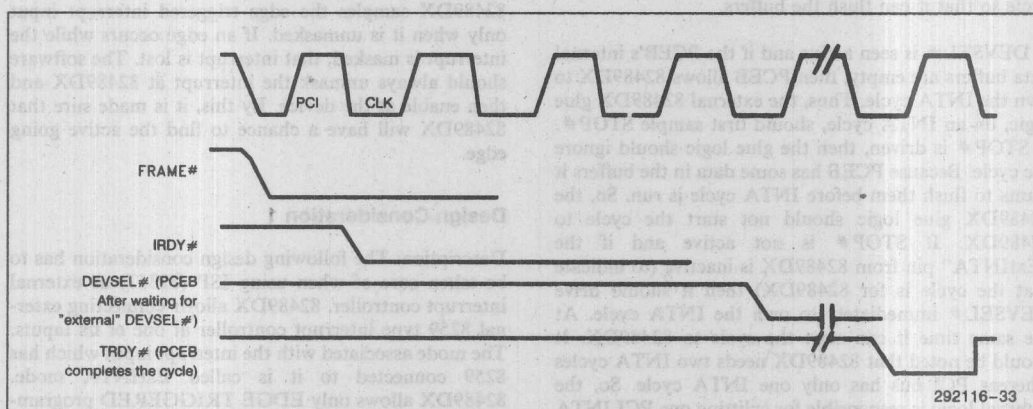
Write x to task priority register to raise priority to x



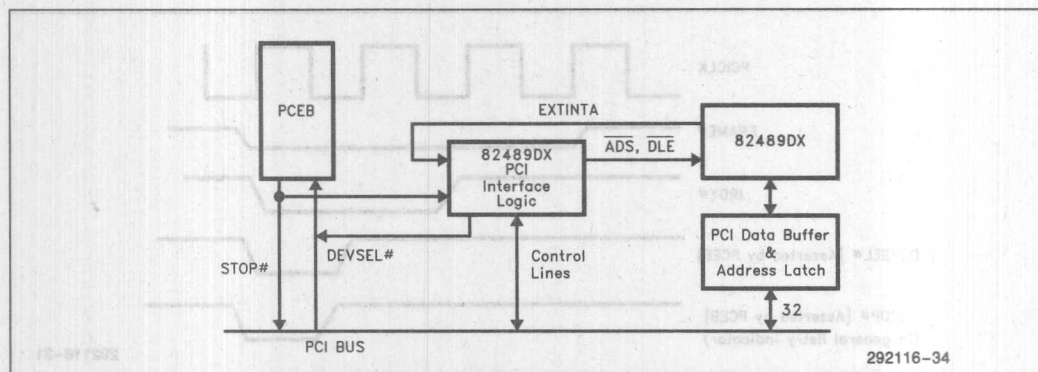
Case 1) Any INTA# Cycle Buffer Management and Retry



**Case 2) Interrupt Acknowledge Cycle
Source of the Interrupt and Vector is 82489DX**



**Case 3) Interrupt Acknowledge Cycle
Source of the Interrupt and Vector is ESC 8259**



82489DX-PCI Interface

82489DX AND PCI-EISA BRIDGE INTEROPERABILITY

82489DX gives performance benefits in multithreaded operating systems. Thus in both uniprocessor and multiprocessor systems 82489DX enhances the system level performance. This is because of its advantage in task priority management. Intel's PCIsset EISA bridge component PCEB (B-stepping onwards) can interoperate with 82489DX. In this section, we will go over the "hook" provided by PCEB to connect 82489DX in a PCI system with some external glue logic.

From the Interrupt acknowledge timings of PCEB (B-stepping onwards) it can be inferred that **whenever the internal data buffers are empty**, on an interrupt acknowledge cycle, PCEB waits one clock cycle so that PCI interface logic for 82489DX can activate DEVSEL#. But, if the internal data buffers are not empty, then PCEB drives STOP# to retry the INTA cycle so that it can flush the buffers.

If DEVSEL# is seen active and if the PCEB's internal data buffers are empty, then PCEB allows 82489DX to own the INTA cycle. Thus, the external 82489DX glue logic, on an INTA cycle, should first sample STOP#. If STOP# is driven, then the glue logic should ignore the cycle. Because PCEB has some data in the buffers it wants to flush them before INTA cycle is run. So, the 82489DX glue logic should not start the cycle to 82489DX. If STOP# is not active and if the "ExtINTA" pin from 82489DX is inactive (to indicate that the cycle is for 82489DX) then it should drive DEVSEL# immediately to own the INTA cycle. At the same time it can start the cycle to 82489DX. It should be noted that 82489DX needs two INTA cycles whereas PCI bus has only one INTA cycle. So, the external logic is responsible for splitting one PCI INTA cycle into two 82489DX INTA cycles back to back but pass only one READY to the system.

During INTA cycle on PCI bus if "ExtINTA" pin is active (to indicate that it is 8259 INTA cycle), then the 82489DX glue logic should not drive DEVSEL#. Thus by finding DEVSEL# inactive, the PCEB will respond to the INTA cycle.

Thus with minimal external glue logic, it is possible to design an APIC based PCI system. Since PCI local bus will improve the I/O performance of the system, APIC will enhance the improvement of the overall system performance in a multithreaded environment.

HARDWARE DESIGN CONSIDERATIONS

Design Consideration 0

Any edge triggered interrupt creating an active edge while the interrupt is masked at 82489DX is lost. The 82489DX samples the edge triggered interrupt input only when it is unmasked. If an edge occurs while the interrupt is masked, that interrupt is lost. The software should always unmask the interrupt at 82489DX and then enable at the device. By this, it is made sure that 82489DX will have a chance to find the active going edge.

Design Consideration 1

Description: The following design consideration has to be taken care of when using ISP (82357) as external interrupt controller. 82489DX allows connecting external 8259 type interrupt controller at one of its inputs. The mode associated with the interrupt input which has 8259 connected to it is called ExtINTA mode. 82489DX allows only EDGE TRIGGERED program-

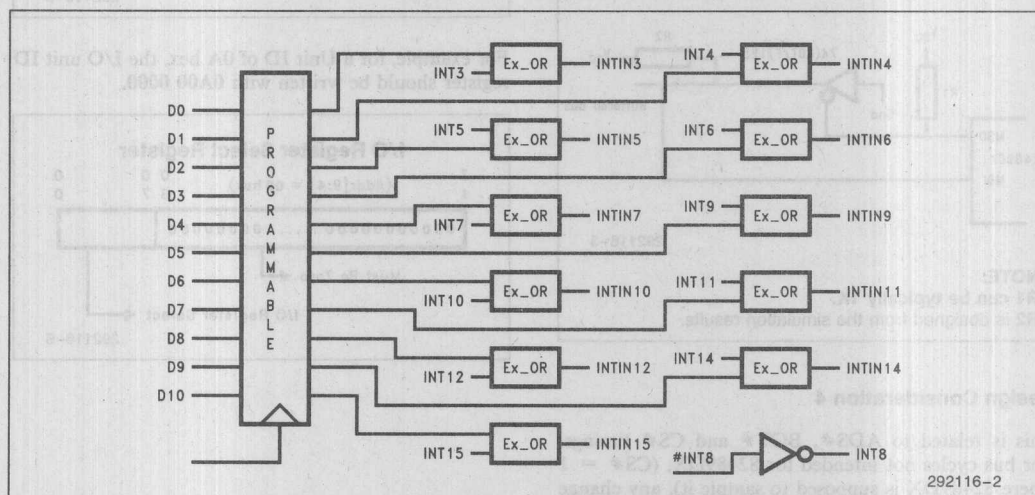
ming option for ExtINTA mode. But in the case of 82357, the INT output from ISP stays high in case more than one interrupt is pending at its inputs. It does not always inactivate its INT output after INTA cycle. This will lead to a situation where ISP keeps the interrupt at high level continuously and waits for INTA cycle. But since 82489DX expects an edge for interrupt sensing (for ExtINTA interrupts) it does not pass the interrupt to CPU and further interrupts are lost. So External circuitry should monitor the end of SECOND CYCLE of INTA cycle and force an inactive state at 82489DX's input. This can be done by ANDing ISP's output with a forced brief low going pulse at the end of second INTA cycle. This will generate an edge for each interrupt at 82489DX's input. For more refined edge generating logic, refer to data book, Order Number 290446.

Design Consideration 2

Description: The following design consideration has to be taken care of when using 82489DX in EISA systems. EISA ISP(82357) chip integrates 8259A. It additionally allows sharing of interrupts. To facilitate this sharing it has a programmable register, ELCR (Edge / Level trigger control register) by which certain interrupt inputs can be programmed as edge (low to high except for RTC) or level (the level is active low). The determination of edge or level is done during initial configuration of EISA system by reading EISA add in boards from the interrupt description data structures. The solution

is to have programmable logic at the interrupt inputs so that 82489DX is compatible with EISA ISP. This will introduce one more register and logic to support this. This should be an 11 bit programmable register and an array of ExOR logic (12 ExOR gates or equivalent PLD). The ISP allows programmability of the following interrupts. It is highly recommended to use the same address of ELCR (and also bit definitions) for this polarity register, if possible, so that when ELCR is written this register will also be written. By this there is no separate programming needed for this polarity register. This will help to maintain compatibility with future APIC implementations which may use the existing ELCR register itself for polarity control. This is true for integrated APIC.

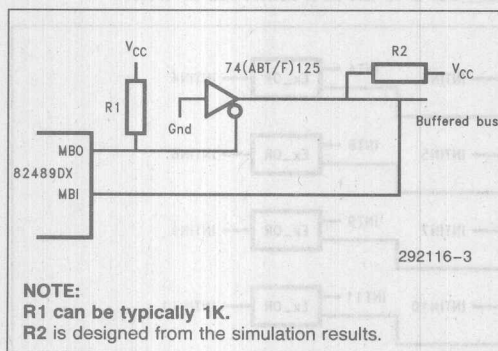
INT3 INT4 INT5 INT6 INT7 INT9 INT10 INT11 INT12 INT14 INT15. In addition to the above 11 interrupts, it fixes INT8 to be active low edge triggered interrupt. INT8 is the only case where it is active low edge triggered type. So the following logic can be used to add programmability in 82489DX based EISA system. Before connecting these 11 interrupt lines directly (#INT8 which is from Real Time Clock is always active low edge triggered. #INT8 can be passed through an inverter since there is no need for programmability) to the 82489DX they should pass through an array of 11 Ex_OR gates. One input of Ex_OR gate connects to the corresponding INT pin and other input connects to a bit of programmable register. The output of Ex_OR gate is connected to 82489DX. The idea of Ex_OR is to use as a controlled inverter.



INTIN are the interrupt inputs to the 82489DX and INT are the system interrupt. The Ex__OR gating register is programmed after EISA configuration is found from add in boards as how these interrupt lines are going to be used in that particular configuration. If a particular input is edge triggered, then the corresponding bit in the register is written with 0. If a particular input is level triggered, then the corresponding bit in the register is written with 1.

Design Consideration 3

I_{CC} bus drive is an open drain bus with drive capacity of 4 mA only. Since data is transmitted at each I_{CC} clock, the "charging" of I_{CC} bus should be fast enough to ensure proper logic level at each clock edge. The I_{CC} bus needs pull up resistors since it is open drain bus. Since the drive is only 4 mA, the pull up resistor value can not be less than 5V/4mA. This being the limit of the resistor value, the length and the characteristics of the I_{CC} trace forces a capacitance value. Both the resistor and capacitance brings a RC time constant to the I_{CC} bus waveform. So, Electrical consideration has to be given to and practice of controlled impedance should be exercised for layout of the I_{CC} bus. The length of the trace should be kept as minimum as possible. If the length of the I_{CC} bus can't be kept less, than say 6 inch, because of mechanical design of the system, the external line drivers should be added to I_{CC} bus and I_{CC} bus should be simulated with the added driver characteristics.



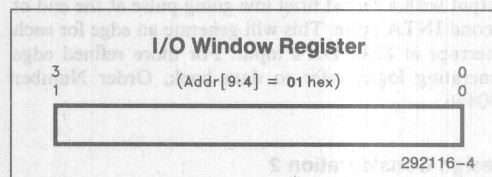
Design Consideration 4

This is related to ADS#, BGT# and CS# timings. For bus cycles not intended for 82489DX, (CS# = 1 where 82489DX is supposed to sample it), any change in CS# line while the ADS# is still active, may erroneously cause a RDY# returned from 82489DX. Anomalous behavior may result if for BGT# ties low cases

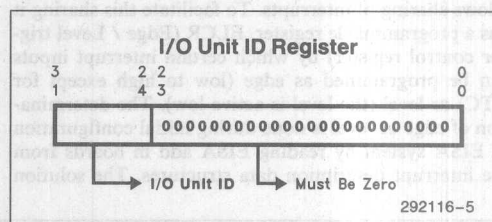
- BGT# goes away just one clock after ADS# or
- ADS# is still active, and CS# changes during this period.

For other cases anomalous behavior results if CS# changes when ADS# is still active. The following considerations are important from timing point of view. Always limit the pulse width of 82489DX ADS# to one CLKIN. Also avoid changing levels on BGT# / CS# line, when ADS# is active for cases being identified as BGT# tied low (BGT# sampled low when ADS# goes active). Also avoid changing levels on CS# line when BGT# is active.

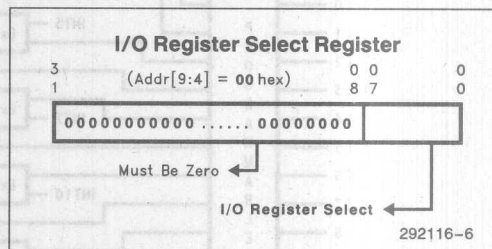
REGISTER PROGRAMMING DETAILS



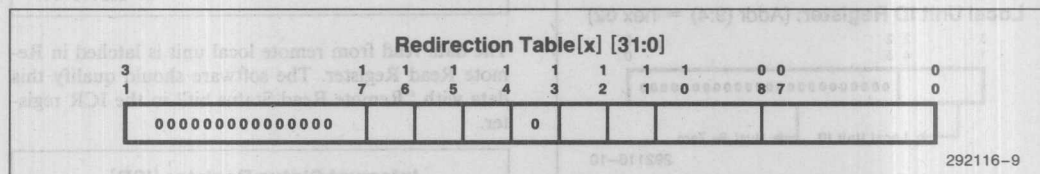
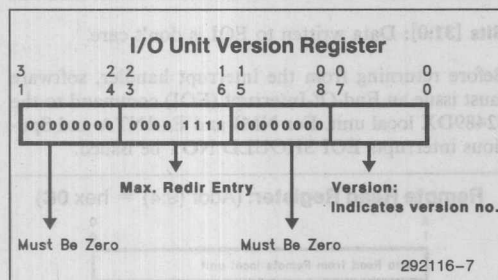
Data access to the register selected by I/O register select Register.



For example, for a Unit ID of 0A hex, the I/O unit ID register should be written with 0A00 0000.



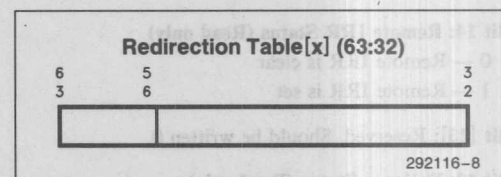
I/O Register Select	Register Selected
00 hex	I/O Unit ID Register
01 hex	I/O Unit Version Register
10 hex	Redirection Table[0] (31:0)
11 hex	Redirection Table[0] (63:32)
12 hex	Redirection Table[1] (31:0)
13 hex	Redirection Table[1] (63:32)
14 hex	Redirection Table[2] (31:0)
15 hex	Redirection Table[2] (63:32)
•	•
•	•
1E hex	Redirection Table[7] (31:0)
1F hex	Redirection Table[7] (63:32)
20 hex	Redirection Table[8] (31:0)
21 hex	Redirection Table[8] (63:32)
•	•
•	•
2E hex	Redirection Table[15] (31:0)
2F hex	Redirection Table[15] (63:32)



I/O Unit Version Register is read only register. It reads as 000F 00XX where XX is version.

Max.Redir Entry: This is equal to the number of interrupt input pins minus 1 of this I/O unit. Read as 15 in 82489DX.

Version: The version number that identifies this version.



Destination: If the destination mode of this entry is "Physical Mode", then the 8 MSB (bits 56 through 63) contain an 82489DX local unit ID.

If logical mode, then all the 32 bits (bits 63 through 32) of the Destination field potentially defines a set of processors.

NOTE:

The same format holds good for Redirection Tables 0 to 15 (x = 0 to 15) for bits 63 to 32.

If the destination is to a local unit with ID, say, 05 in physical mode, then the redirection table [63:32] should be programmed as hex 0500 0000.

Redirection Table [63:32] should be programmed first before programming Redirection Table [31:0].

Bits [31:17]: Reserved. Should be written 0.

Bit 16: MASK

- 0 — Not masked
- 1 — Masked

Bit [15]: TRIGGER MODE

- 0 — Edge Triggered
- 1 — Level Triggered

Bit 14: Remote IRR Status (Read only)

- 0 — Remote IRR is clear
- 1 — Remote IRR is set

Bit [13]: Reserved. Should be written 0.

Bit 12: Delivery Status (Read only)

- 0 — Idle
- 1 — Send Pending

Bit [11]: Destination Mode

- 0 — Physical
- 1 — Logical

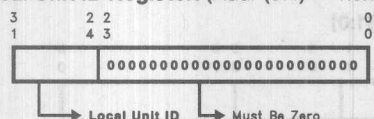
Bits [10:8] Delivery Mode

- 000: Fixed
- 001: Lowest Priority
- 100: NMI
- 101: Reset
- 111: ExtINTA

Bits [7:0] Vector

Vector for this interrupt

Local Unit ID Register: (Addr (9:4) = hex 02)

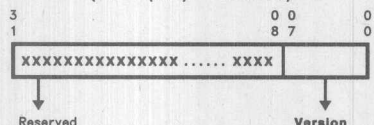


292116-10

For example, for a local Unit ID of 0A hex, the local unit ID register should be written with 0A00 0000.

Local Unit Version Register

(Addr (9:4) = hex 03)

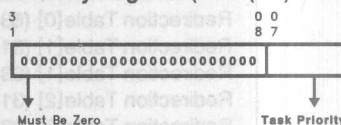


292116-11

Local Unit Version Register is read only register. It reads as 0000 00YY where YY is version number.

Bits [7:0] Version: The version number that identifies this version.

Task Priority Register (Addr (9:4) = hex 08)

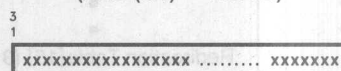


292116-12

Bits [0:7] Task Priority: Should be written with task priority.

End Of Interrupt (EOI) Register

(Addr (9:4) = hex 0B)



292116-13

Bits [31:0]: Data written to EOI is don't care.

Before returning from the interrupt handler, software must issue an End-Of-Interrupt (EOI) command to the 82489DX local unit. For NMI and ExtINTA and Spurious interrupts EOI SHOULD NOT be issued.

Remote Read Register: (Addr (9:4) = hex 0C)

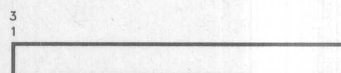


292116-14

The data read from remote local unit is latched in Remote Read Register. The software should qualify this data with "Remote Read Status bit" in the ICR register.

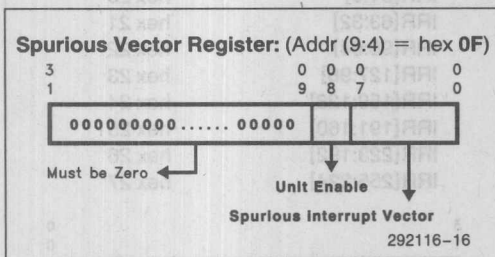
Interrupt Status Register [ISR]:

Register	Address[9:4]
ISR[31:0]	hex 10
ISR[63:32]	hex 11
ISR[95:64]	hex 12
ISR[127:96]	hex 13
ISR[159:128]	hex 14
ISR[191:160]	hex 15
ISR[223:192]	hex 16
ISR[255:224]	hex 17



292116-15

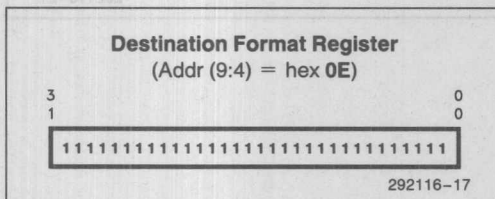
Interrupt Status Register is read only. It marks the interrupts that have been delivered to the processor and waiting for EOI.



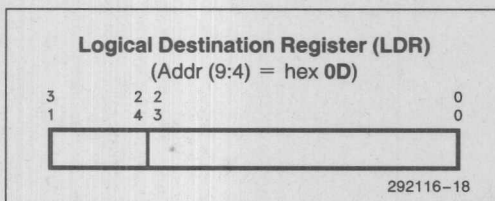
Bits [31:09]: Reserved bits. Must be zero.

Bit 8: Unit Enable: When this bit is 0, the local unit is disabled with regard to transmit and responding messages on ICC bus. It only responds to messages with delivery mode set to "Reset". Reading a 0 at this bit indicates that the unit is disabled. When a 1 is written to the bit, the local unit is enabled for both transmitting and receiving messages. **Once enabled, it should not be disabled by software. Only further resets can take the unit into disabled condition.**

Bits [7:0] Spurious Interrupt Vector: For future compatibility, the bits [3:0] should be written with 1111. A spurious interrupt service routine should be existing in the address corresponding to the spurious interrupt vector.



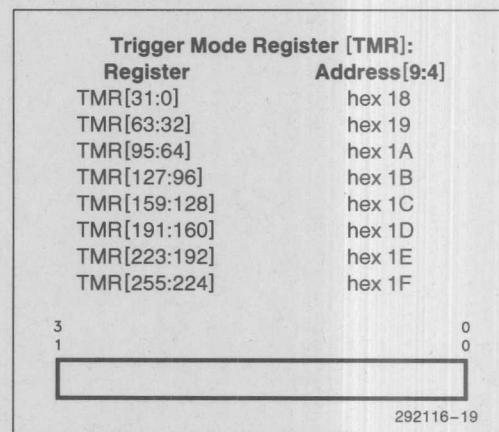
The destination format register enables logical addressing by specifying the bit map in logical destination register. For future compatibility, all the 32 bits of Destination Format Register should be 1.



Each local unit can be addressed either physically using physical ID or logically using logical destination register. In physical addressing, either only one local unit can be addressed at a time or broadcast to all local units can be done. In logical addressing, a group of local units can be addressed through bit mapping in destination addressing and the logical destination register.

For future compatibility, bits [0:23] of the logical destination register and bits [0:23] of the destination address in the message should be zero. Bits 24 through 31 of destination information in the interrupt message received are interpreted as decoded field. This field is compared against the logical destination register of the local unit. If there is a bit match (i.e., at least one of the corresponding pair of bits of the destination field and LDR match) that unit is selected for interrupt delivery. Each bit position in the destination field corresponds to an individual Local unit. This scheme allows the specification of arbitrary groups of local units by setting the member's bits to 1, but allows a maximum of 8 local units in a system since only bits 24 through 31 (of the Logical Destination Register and logical destination address in interrupt message) are used. Broadcast to all is achieved by setting all 8 bits of destination to ones. This selects all local units in the system.

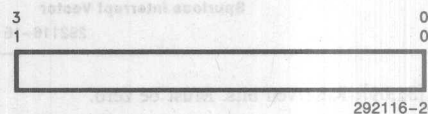
In a very large multiprocessor system where future compatibility is not a main problem, all the 32 bits of the Logical Destination Register and all the 32 bits of the destination address in the interrupt message can be used as a bit map to address the processors. When message addresses the destination using logical addressing scheme, the local unit compares the logical address in the interrupt message with its own logical Destination Register. Thus it is possible to support 32 processors in logical addressing mode.



If a bit corresponding to an interrupt vector number is 0, then it is assumed as edge triggered interrupt. For edge triggered interrupt, the corresponding IRR bit is automatically cleared when interrupt service starts. If 1 (level triggered) this is not the case. Instead, the source 82489DX (source I/O unit or Source Local Unit) must explicitly request the IRR bit be cleared (upon deassert of the interrupt input pin or upon sending an appropriate interprocessor interrupt). Upon acceptance of interrupt, the TMR bit is cleared for edge triggered interrupts and set for level triggered interrupts. This information was carried in the accepted interrupt message. The source 82489DX I/O unit also tracks the state of the destination unit's IRR bit (Remote IRR bit in the redirection table). When a level triggered interrupt input is deasserted, the source 82489DX I/O unit detects the discrepancy between the input pin state and the Remote IRR, and automatically sends a message telling destination 82489DX to clear IRR for the interrupt.

Interrupt Request Register [IRR]:

Register	Address [9:4]
IRR[31:0]	hex 20
IRR[63:32]	hex 21
IRR[95:64]	hex 22
IRR[127:96]	hex 23
IRR[159:128]	hex 24
IRR[191:160]	hex 25
IRR[223:192]	hex 26
IRR[255:224]	hex 27



It contains the active interrupt requests that have been accepted, but not yet dispensed by this 82489DX local unit. A bit in IRR is set when 82489DX local unit accepts the interrupt. When TMR is 0, it is cleared when the interrupt is serviced; when TMR is 1, it is cleared when the 82489DX local unit receives a message to clear it.

Interrupt Command Register [31:0] (Addr [9:4] = 30 hex)

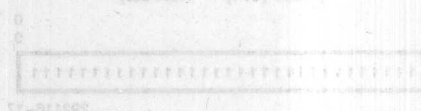


292116-21

Register	Address [9:4]
TMR[31:0]	hex 18
TMR[63:32]	hex 19
TMR[95:64]	hex 1A
TMR[127:96]	hex 1B
TMR[159:128]	hex 1C
TMR[191:160]	hex 1D
TMR[223:192]	hex 1E
TMR[255:224]	hex 1F



Destination Format Register (DFR)



Logic Destination Register (LDR)



Bits [31:20]: Reserved. Should be written 0.

Bits [19:18]: Destination Shorthand. This field indicates whether a shorthand notation is used to specify the destination of the interrupt and if so, which shorthand is used. Destination shorthands do not use the 32-bit Destination field, and can be sent by software with a single 32-bit write to the 82489DX's interrupt command register. Shorthands are defined for the following cases: Software self interrupt, interrupt to all processors in the system including the sender, interrupts to all processors in the system excluding the sender.

00: (dest field) means that no shorthand is used. The destination is specified in the 32-bit Destination field in the second word (bits 32 to 63) of the interrupt control register.

01: (self) means that the current local unit is the single destination of the interrupt. This is useful for software interrupts. The destination field in the interrupt command register is ignored. RESET assert Delivery mode should not be used with self destination. Only FIXED delivery mode should be used with SELF.

10: (all incl. self) means that the interrupt is to be sent to "all" processors in the system including the processor sending the interrupt. The 82489DX will broadcast a message with destination unit ID field set to all ones. RESET assert Delivery mode should not be used with "all incl. self" destination.

11: (all excl. self) means that the interrupt is to be sent to all processors in the system excluding the processor sending the interrupt. The 82489DX will broadcast a message with destination unit ID field set to all ones.

Bits [17:16]: Remote Read Status. This field indicates the status of the data contained in the Remote Read register. This field is read only to software. Whenever software writes to the interrupt command register using Delivery mode "Remote Read" the Remote Read Status becomes "in progress" (waiting for the remote data to arrive). The remote 82489DX local unit is expected to respond in a fixed amount of time. If the remote 82489DX local unit is unable to do so, then the remote read status becomes "invalid". If successful, the Remote Read status resolves to "Valid". Software should poll this field to determine completion and success of the Remote Read command.

00: (invalid): The content of the Remote Read register is invalid. This is the case when after a Remote Read command is issued and the remote 82489DX Local unit was unable to deliver the Register content in time.

01: (in progress): a remote read command has been issued and this 82489DX is waiting for the data to arrive from remote 82489DX local unit

10: (valid): the most recent Remote Read command has completed and the remote read register content is valid.

11: reserved.

Bit [15]: TRIGGER MODE

0 — Edge Triggered

1 — Level Triggered

Software should use this bit in conjunction with Level Assert/Deassert to generate interrupts that behave as edges or levels. **For future compatibility, send ICR messages only in edge triggered mode.**

Bit [14]: LEVEL. Software should use this bit in conjunction with the Trigger mode bit when issuing an inter-processor interrupt to simulate assertion/deassertion of level sensitive interrupts.

To assert: Trigger mode = 1 and Level = 1.

To deassert: Trigger mode = 1 and Level = 0.

For example, a message with Delivery mode of "Reset", a trigger mode of "Level", and Level bit of 0 deasserts reset to the processor of the addressed 82489DX Local unit(s). As a side effect, this will also cause all 82489DX to reset their Arbitration ID to their unit ID. (The Arb ID is used for tie breaking in lowest priority arbitration.) **For future compatibility, only edge triggering should be used in ICR.**

Bit [13]: Reserved. Should be written 0.

Bit [12]: Delivery Status (Read only)

0 — Idle

1 — Send pending

Delivery status is software read-only. Software can read to find out if the current interrupt has been sent, and the Interrupt command register is available to send the next interrupt. If the interrupt command register is overwritten before the Delivery status is "idle", then the destiny of that interrupt is undefined; the interrupt may have been lost.

Bit [11]: Destination Mode

0 — Physical

1 — Logical

In physical mode, a destination 82489DX is identified by its Local Unit ID. Bits 56 through 63 (8 MSB of the destination field) specify the 8-bit 82489DX Local unit ID.

In logical mode, destinations are identified by matching on Logical Destination under the control of the Destination Format Register in each Local 82489DX. The 32-bit Destination field is the logical destination. For future compatibility, use only bits [31:24] of the logical destination address. Bits [23:0] should be zero.

Bits [10:8]: Delivery Mode

- 000: (Fixed) means deliver the signal on the INT pin of all processors listed in the destination. Trigger mode for "fixed" Delivery Mode can be edge or level.
- 001: (Lowest Priority) means deliver the signal on the INT pin of the processor that is executing at the lowest priority among all the processors listed in the specified destination; Trigger mode for "lowest priority" Delivery mode can be edge or level.
- 011: (Remote Read) is a request to a remote 82489DX local unit to send the value of one of its registers over the ICC bus. The register is selected by providing its address in the vector field. The register value is latched by the requesting 82489DX and stored in the Remote Register where it can be read by the local processor. A Delivery Mode of "Remote Read" requires an "Edge" Triggered mode.
- 100: (NMI) means deliver the signal on the NMI pin of all processors listed in the destination. Vector information is ignored. A delivery mode equal to "NMI" requires a "LEVEL" Trigger mode.
- 101: (Reset) means deliver the signal to all processors listed in the destination by asserting/deasserting the 82489DX local unit's PRST output pin. All

addressed 82489DX local units will assume their reset state but preserve their ID. One side effect of a message with Delivery mode equal to "Reset" that results in a deassert of reset is that all Local Units (whether listed in the destination or not) will reset their lowest-priority tie breaker arbitration ID to their Local unit ID. A delivery mode of "Reset" requires a "level" Trigger mode. "Reset" should not be used with "Self" or "all incl.Self" Shorthand mode since it will leave the system in non-recoverable reset state. If "RESET" is used with "all exc.Self" mode, software should make sure that only one CPU executes this instruction in an MP system.

Delivery mode options 010,110,111 are Intel reserved. They should not be used.

Bits [7:0] Vector. The vector identifies the interrupt being sent. If the Delivery mode is "Remote Read", then the Vector field contains the address of the register to be read in the remote 82489DX's Local unit.

NOTE:

In cases where Destination field in Interrupt Command Register [63:32] is used, Interrupt Command Register [31:0] should be programmed only **AFTER** programming Interrupt Command Register [63:32], since writing to [31:0] will start sending the message.

The following are the control words for interrupt command register [31:0] for different modes. The interrupt vector, for example, is illustrated with AA hex. In the remote Read request command RR in the vector field specify address of the register to be read. The XX in the vector field means the vector is don't care.

CONTROL WORD	PHYSICAL Destination Mode	LOGICAL Destination Mode
Fixed INT, Edge triggered int, <i>dest. field specified</i>	0000 00AA hex	0000 08AA hex
Lowest priority INT, Edge trigg. int, <i>dest. field specified</i>	0000 01AA hex	0000 09AA hex
Remote Read (only Edge Triggered), <i>dest. field specified</i>	0000 03RR hex	NOT SUPPORTED
NMI (Only Level) Level ASSERT, <i>dest. field specified</i>	0000 C4XX hex	0000 CCXX hex
NMI (Only Level) Level DEASSERT, <i>dest. field specified</i>	0000 84XX hex	0000 8CXX hex
Reset (Only Level) Level ASSERT, <i>dest. field specified</i>	0000 C5XX hex	0000 CDXX hex
Reset (Only Level) Level DEASSERT, <i>dest. field specified</i>	0000 85XX hex	0000 8DXX hex
Fixed INT, Edge triggered int, <i>Self</i>	0004 00AA hex	0004 08AA hex
Fixed INT, Edge trigg. int, <i>All inclusive Self</i>	0008 00AA hex	0008 08AA hex
Lowest priority INT, Edge trigg. int, <i>All inclusive Self</i>	0008 01AA hex	0008 09AA hex
NMI, Level ASSERT, <i>All inclusive Self</i>	0008 C4XX hex	0008 CCXX hex
NMI, Level DEASSERT, <i>All inclusive Self</i>	0008 84XX hex	0008 8CXX hex
Reset, Level DEASSERT, <i>All inclusive Self</i>	0008 85XX hex	0008 8DXX hex
Fixed INT, Edge trigg. int, <i>All exclusive self</i>	000C 00AA hex	000C 08AA hex
Lowest priority INT, Edge trigg. int, <i>All exclusive self</i>	000C 01AA hex	000C 09AA hex
NMI, Level ASSERT, <i>All exclusive Self</i>	000C C4XX hex	000C CCXX hex
NMI, Level DEASSERT, <i>All exclusive Self</i>	000C 84XX hex	000C 8CXX hex
Reset, Level ASSERT, <i>All exclusive Self</i>	000C C5XX hex	000C CDXX hex
Reset, Level DEASSERT, <i>All exclusive Self</i>	000C 85XX hex	000C 8DXX hex

Interrupt Command Register [63:32]

(Addr [9:4] = 31 hex)

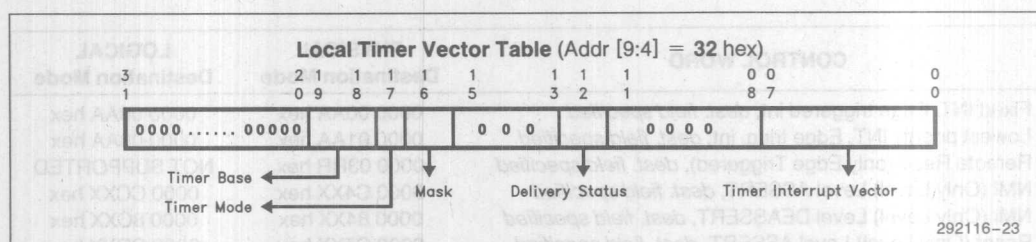
Bits [63:32]

292116-22

Bits [63:32] Destination

This field is only used when the Destination Shorthand field is set to "Destination Field". If Destination field is physical mode, then the 8 MSB contain an Destination Unit ID. If logical mode, the full 32-bit Destination field contains the logical address. This register should be programmed for proper destination before programming Interrupt Command Register [31:0]. If the destination to a local unit with ID, say, 05 in physical mode, then the Interrupt Command Register [63:32] should be programmed as hex 0500 0000.

Control Word	CLKIN Input (Base 0)	TMBASE Input (Base 1)	Divisor Input (Base 2)
PERIODIC timer MASK cleared	0005 00AA hex	0005 00AA hex	000A 00AA hex
PERIODIC timer MASK set	0003 00AA hex	0007 00AA hex	000B 00AA hex
ONE SHOT timer MASK cleared	0006 00AA hex	0004 00AA hex	0008 00AA hex
ONE SHOT timer MASK set	0007 00AA hex	0005 00AA hex	0009 00AA hex



Bits [31:20] Reserved. Should be written Zero.

Bits [19:18] Timer Base: This field selects the time base input to be used by timer.

- 00: (Base 0): Uses "CLKIN" as input.
- 01: (Base 1): Uses "TMBASE".
- 10: (Base 2): Uses the output of the divider (Base 2).

Bit 17: Timer Mode: This field indicates the operation mode of timer.

- 0 — ONE-SHOT;
- 1 — PERIODIC

In *ONE-SHOT*, the current count register remains at Zero after the timer reaches zero and software needs to reassign the timer's initial count register to rearm the timer.

In *PERIODIC* mode, when the timer reaches zero, the Current Count Register is automatically reloaded with the value in the initial Count Register, and the timer counts down again.

Bit 16: Mask: This bit serves to mask timer interrupt generation.

- 0 — Not masked;
- 1 — Masked.

Bits [15:13] Reserved. Should be written Zero.

Bit [12] Delivery Status: Delivery status indicates the current status of the delivery status of this interrupt.

- 0 — IDLE means that there is currently no activity for this interrupt.
- 1 — SEND PENDING indicates that the interrupt has been injected, but its delivery is temporarily held up by other recently injected interrupts that are in the process of being delivered; Delivery status is software read only.

Bits [11:8] Reserved. Should be written Zero.

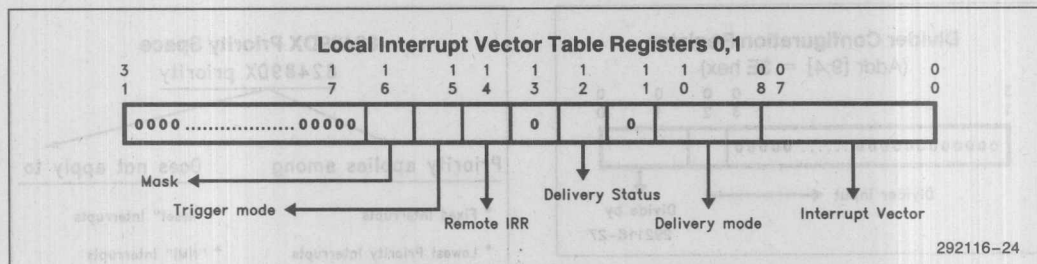
Bits [7:0] Timer Interrupt vector: This is the 8-bit interrupt vector to be used when timer generates an interrupt.

NOTE:

TIMER interrupts are always treated as EDGE triggered interrupts.

The following is the control word for various modes to be used in Local Timer Vector Table. For illustration purpose, the interrupt vector for Timer is shown as AA hex.

Control Word	CLKIN Input (Base 0)	TMBASE Input (Base 1)	Divider Input (Base 2)
PERIODIC timer, MASK cleared	0002 00AA hex	0006 00AA hex	000A 00AA hex
PERIODIC timer, MASK set	0003 00AA hex	0007 00AA hex	000B 00AA hex
ONE SHOT timer, MASK cleared	0000 00AA hex	0004 00AA hex	0008 00AA hex
ONE SHOT timer, MASK set	0001 00AA hex	0005 00AA hex	0009 00AA hex



Register	Address [9:4]
Local Int0 Vector table register	35 hex
Local Int1 Vector table register	36 hex

The same format applies to both Local Int0 and Local Int1 registers.

Bits [31:17]: Reserved: Must be Zero.

Bit 16: MASK:

- 0 — enables interrupt by clearing mask
- 1 — masks the interrupt.

Bit 15: Trigger mode:

- 0 — Edge Triggered
- 1 — Level Triggered

Bit 14: Remote IRR: This bit is used for level triggered local interrupts. Its meaning is undefined for edge triggered interrupts. Remote IRR mirrors the interrupt's IRR bit of this local unit. Remote IRR is software read only.

Bit 13: Reserved. Must be Zero.

Bit 12: Delivery Status: Software read only. Indicates the current status of the delivery of this interrupt.

- 0 — IDLE means that there is currently no activity for this interrupt.
- 1 — Send Pending indicates that the interrupt has been injected, but its delivery is temporarily held up by the recently injected interrupts that are in the process of being delivered.

Bit 11: Reserved. Must be Zero.

Bits [10:8]: Delivery mode

- 000 — Fixed INT
- 100 — NMI
- 111 — ExtINTA

All other options of Bits [10:8] are reserved. Should not be used.

Bits [7:0]: Vector: This is the interrupt vector to use when generating interrupt for this entry.

The following are the control words for local interrupt [0 as well as 1] vector tables for different modes. The interrupt vector, for example, is illustrated with AA hex. The XX in the vector field means the vector is don't care.

Interrupt Option	Control Word
Fixed INT, Edge triggered	0000 00AA hex
Fixed INT, Level trigg. int	0000 80AA hex
NMI (Only Level)	0000 84XX hex
ExtINTA (Only Edge)	0000 07XX hex

Initial Count Register (Addr [9:4] = 38 hex)

Bits [31:0]

292116-25

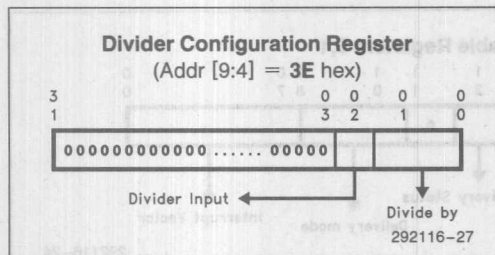
Bits [31:0] Initial Count: Software writes to this register to set the initial count for timer. This register can be written at any time. When written, the value is copied to the current count Register and countdown starts or continues from there. The initial count register is read-write by software.

Current Count Register (Addr [9:4] = 39 hex)

Bits [31:0]

292116-26

Bits [31:0] Current Count: This is the current count of timer. It is read only by software and can be read any time.



Configuration	Control Word
Divide CLKIN by 2	0000 0000 hex
Divide CLKIN by 4	0000 0001 hex
Divide CLKIN by 8	0000 0002 hex
Divide CLKIN by 16	0000 0003 hex
Divide TMBASE by 2	0000 0004 hex
Divide TMBASE by 4	0000 0005 hex
Divide TMBASE by 8	0000 0006 hex
Divide TMBASE by 16	0000 0007 hex

Bits [31:3] Reserved. Must be Zero.

Bit [2]: Divider Input: Selects whether divider's input connects to the 82489DX local unit's CLKIN pin or TMBASE.

0 — means the divider takes its input signal from CLKIN.

1 — means use TMBASE

Bits [1:0]: Divide by: Selects by how much the divider divides.

00 — divide by 2

01 — divide by 4

10 — divide by 8

11 — divide by 16

Programming Guidelines

A) Modes of Interrupt in 82489DX:

Trigger Mode	Delivery Mode				
	Fixed Destination	Lowest Priority Delivery	NMI	Reset	ExtINT
Edge	✓	✓			✓
Level	✓	✓	✓	✓	

NOTE:

- RESET delivery mode should not be used for Local Interrupts.
- EOI should not be issued for NMI and ExtINT delivery mode.

82489DX Priority Space

82489DX priority

Priority applies among

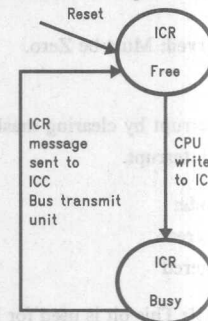
- Fixed Interrupts
- Lowest Priority Interrupts
- "Processor Task priority"

Does not apply to

- "Reset" Interrupts
- "NMI" Interrupts
- "ExtINT" Interrupts
- Spurious Interrupts

292116-29

Interrupt Command Register State Diagram



292116-30

NOTE:

*Software should check busy flag of ICR before writing interprocessor message.

CONCLUSION

82489DX has simple and powerful programming model. It has programmable priority and it supports task priority in the light of interrupt priority. It reduces the SPL() overhead which is very useful in uniprocessor system. The system performance is improved by using interrupt priority model to prioritize interrupts and by using task priority register for SPL() calls. It provides an easy migration path from 8259 by providing ExtINTA mode for DOS compatibility.

APPLICATION NOTE

3

An Example Memory Subsystem for the Pentium™ Processor

RAHEEL GHAZNAVI
TECHNICAL MARKETING

November 1993

PRELIMINARY

3-27

An Example Memory Subsystem for the Pentium™ Processor

CONTENTS	PAGE	CONTENTS	PAGE
1.0 INTRODUCTION	3-30	7.2 Write Cycle Timing	3-56
2.0 Pentium™ PROCESSOR BUS CHARACTERISTICS	3-30	7.2.1 Pipelined Write Cycles	3-56
2.1 Bus Cycle Mix	3-31	7.2.2 Burst Write Cycles	3-57
2.2 Bus Utilization	3-32	7.3 Pipelined Read-Write Cycles	3-58
3.0 MEMORY SUBSYSTEM PERFORMANCE	3-33	7.4 Pipelined Write-Read Cycles	3-59
3.1 Overall Performance	3-33	7.5 Snoop Cycles	3-60
3.2 Impact of Wait States on Performance	3-34	8.0 CRITICAL TIMING CONSIDERATIONS	3-61
3.2.1 Read Cycle Performance	3-34	8.1 DRAM Access Timing	3-61
3.2.2 Write Cycle Performance	3-36	8.2 Controller Logic Timing	3-62
3.3 Effects of DRAM Interleaving on Performance	3-37	9.0 SUMMARY	3-63
3.4 Effects of DRAM Paging on Performance	3-38	APPENDIX A: PERFORMANCE SIMULATION METHODOLOGY	3-64
3.5 Effects of Memory Bus Pipelining	3-39	APPENDIX B: PARTS LIST	3-65
3.6 Memory Bus Width and Bus Speed	3-41	APPENDIX C: PLD CODES AND SCHEMATICS	3-66
4.0 AN EXAMPLE SYSTEM ARCHITECTURE OVERVIEW	3-42	FIGURES	
5.0 DRAM SUBSYSTEM OVERVIEW	3-43	Figure 1 Bus Cycle Mix for the Pentium™ Processor	3-31
5.1 DRAM Organization	3-45	Figure 2 Overall Performance of the Pentium™ Processor Memory Subsystem Design Example ...	3-34
5.2 Address Path Logic	3-46	Figure 3 Effects of Burst Read Cycle Timing on Performance	3-35
5.3 Data Path Logic	3-48	Figure 4 Effects of Read Lead-Off Cycle Timing on Performance	3-35
5.4 Control Logic	3-49	Figure 5 Effects of Write Cycle Timing on Performance	3-36
6.0 BUS CONVERSION LOGIC	3-51	Figure 6 Effects of Burst Write Cycle Timing on Performance	3-37
6.1 Synchronization of Control Signals	3-52	Figure 7 Effects of DRAM Interleaving on Performance	3-38
7.0 MEMORY CYCLE TIMING	3-52	Figure 8 Effects of DRAM Paging on Performance	3-39
7.1 Read Cycle Timing	3-53	Figure 9 Effects of Memory Bus Pipelining on Performance	3-40
7.1.1 Non-Pipelined Read Cycle After Reset	3-53	Figure 10 Effects of Bus Width on Performance	3-41
7.1.2 Pipelined Read Page Hit Cycle	3-54		
7.1.3 Pipelined Read Page Miss Cycle	3-55		

CONTENTS	PAGE
FIGURES	
Figure 11 Effects of Bus Speed on Performance	3-42
Figure 12 System Architecture Overview	3-43
Figure 13 Block Diagram of the Memory Subsystem	3-44
Figure 14 Organization of the DRAM Array	3-45
Figure 15 Address Path Logic	3-46
Figure 16 DRAM Page Address Comparison Logic	3-47
Figure 17 Data Path Logic	3-49
Figure 18 DRAM Subsystem Control Logic	3-50
Figure 19 Address/Data Bus Conversion Logic	3-51
Figure 20 Synchronization of ADS# from 66-MHz Bus to 33-MHz Bus ...	3-52
Figure 21 DRAM Read Cycle After RESET	3-54
Figure 22 Pipelined Burst Read Page Hit	3-55
Figure 23 Pipelined Read Page Miss Cycle	3-56
Figure 24 Pipelined Write Cycle	3-57
Figure 25 Burst Write Cycle	3-58
Figure 26 Write Cycle Pipelined into a Burst Read Cycle	3-59

[illegible]

1.0 INTRODUCTION

This application note discusses memory subsystem design for the Pentium™ microprocessor. It first illustrates trade-offs in memory subsystem design for a Pentium processor-based system. It then provides a design example of a high performance DRAM subsystem to highlight some of the features of the Pentium processor. Also covered in the example design is the bus conversion logic to interface the Pentium processor memory subsystem with a half speed, 32-bit, Intel486™ CPU-based bus to allow interface to an existing chip set. The example is intended to provide guidelines towards memory subsystem design for the Pentium processor. It is not necessarily the optimal cost/performance solution.

The Pentium processor bus characteristics presented in Section 2.0 and the data provided in the memory subsystem performance section is derived from an Intel internal performance simulator. Based on previous experience with the Intel486 CPU, the simulator has proven to be highly accurate (<5% margin of error compared to the actual hardware configuration it simulates) in simulating the CPU, cache and memory performance. The performance simulation methodology is provided in Appendix A. Also provided in Appendix A is the contact information for obtaining a simulator and Pentium processor models to reproduce or run variations of the data presented in this report.

The discussion in subsequent sections of this application note assumes familiarity with CPU, cache, and memory system architectures. For further reference information on Pentium processor and other related documentation, refer to the following:

1. *Intel486™ Microprocessor-Based 82350 EISA Design Guide*, Order No. 296504-001
2. *82350 EISA Chip Set Data Book*, Order No. 290220-003
3. *Pentium™ Processor Data Book*, Order No. 241428-001

2.0 Pentium™ PROCESSOR BUS CHARACTERISTICS

The internal cache structure of the CPU significantly affects the amount of external bus traffic as it filters the different reads, writes and prefetches before they reach the memory bus. The amount of bus traffic and the bus cycle mix is important to consider when designing an efficient memory subsystem. Table 1 details the Pentium processor bus characteristics for different applications.

Table 1. Pentium™ Processor Bus Characteristics

	UNIX (SPEC1)	DOS (AutoCAD)	WINDOWS (Excel)	WINDOWS (Word)	WINDOWS (Clip)
# of Instructions Simulated	5.04 Million	3.98 Million	1.82 Million	1.26 Million	1.47 Million
L1 Data Cache					
Overall Hit Rate	93.2%	94.4%	83.8%	89.9%	97.2%
Read Hit Rate	96.5%	99.0%	89.7%	93.6%	97.2%
Write Hit Rate	86.8%	88.2%	73.6%	82.0%	97.2%
L1 Instruction Cache					
Overall Hit Rate	95.1%	89.6%	84.0%	92.1%	94.5%
Bus Cycle Mix					
Ratio Data Reads	20.8%	6.2%	23.7%	28.8%	24.7%
Ratio Code Prefetch	32.7%	40.1%	35.8%	25.6%	43.9%
Ratio Data Writes	40.0%	50.8%	35.6%	38.4%	16.6%
Ratio Write-backs	6.5%	2.9%	4.9%	7.2%	14.8%
Memory Bus Utilization(1)	17.0%	14.9%	28.0%	17.8%	19.8%

NOTE:

1. For Zero Wait State Memory System

For the bus statistics presented here and for the performance analysis in the later sections, five different applications were used from DOS, UNIX, and Windows environments. For simulation purposes, traces from each of the five different applications were fed through an Intel Internal (CPU-Cache-Memory) Performance Simulator to extract the desired information. The application traces are hardware/software generated and consist of random samples of instructions taken while the actual programs were running (see Appendix A). These applications are described as following:

DOS

AutoCAD—Auto Desk's AutoCAD program computing and displaying a drawing.

UNIX

Spec1—A mixture of integer SPEC89 benchmark suite programs running concurrently.

Windows

Excel (Spreadsheet)—Microsoft Excel for Windows running a spreadsheet calculation.

Word (Word Processing)—Microsoft Word for Windows converting a document for import.

Clip (Graphics)—Clipping of graphics and text to different screen regions under Windows.

2.1 Bus Cycle Mix

The Bus Cycle Mix for the Pentium processor is illustrated in Figure 1. The result represents an average of the bus cycle mix for the five applications shown in Table 1.

On the Pentium processor, the external bus features a write-back cache protocol to support its internal data cache. This write-back cache architecture significantly reduces the amount of write cycle traffic on the memory bus compared to a write-through cache architecture, as on the Intel486 CPU. However, the number of write cycles coming out on the bus and the ratio of writes to reads is still dependent on the particular application. The average data write-backs, shown in Figure 1, is the number of write cycles that occur as a result of a read miss in the L1 cache.

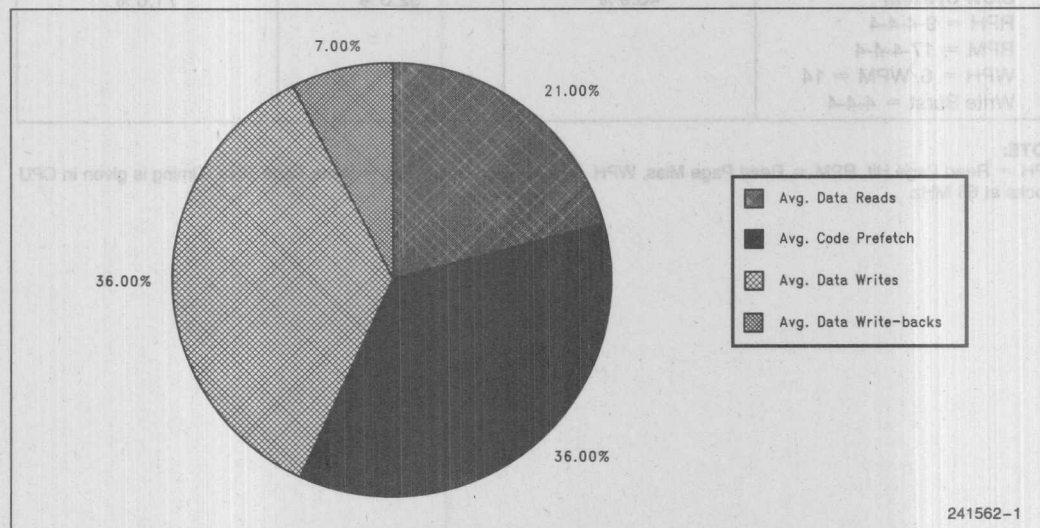


Figure 1. Bus Cycle Mix for the Pentium™ Processor

2.2 Bus Utilization

Table 1 showed the L1 hit rates for different applications. The higher the hit rate in the L1 cache, the less amount of time the CPU has to spend on the external bus. Bus Utilization refers to the amount of time, out of the total time taken to execute the application, that the CPU has to spend executing cycles on the external bus. Bus utilization is not only dependent on the application but also on the memory subsystem. Faster memory sys-

tems allow the bus cycles to complete faster and therefore reduce the CPU bus utilization. Table 2 shows the Pentium processor bus utilization for three different memory subsystems.

As seen from Table 2, the memory bus utilization increases as the memory gets slower; also since the DOS application (AutoCAD) has a higher hit rate in the L1 cache compared to the other applications, its bus utilization numbers are also lower than the rest.

Table 2. Pentium™ Processor Memory Bus Utilization for Three Different Memory Subsystems

	DOS (AutoCAD)	UNIX (Spec1)	Windows (Excel)
Zero Wait State RPH = 2-1-1-1 RPM = 2-1-1-1 WPH = 2, WPM = 2 Write Burst = 1-1-1	14.9%	17.0%	28.0%
Fast System RPH = 5-2-2-2 RPM = 11-2-2-2 WPH = 3, WPM = 7 Write Burst = 2-2-2	28.7%	35.0%	52.7%
Slow System RPH = 9-4-4-4 RPM = 17-4-4-4 WPH = 6, WPM = 14 Write Burst = 4-4-4	43.9%	52.0%	71.6%

NOTE:

RPH = Read Page Hit, RPM = Read Page Miss, WPH = Write Page Hit, WPM = Write Page Miss. Timing is given in CPU clocks at 66 MHz.

3.0 MEMORY SUBSYSTEM PERFORMANCE

The performance of the memory subsystem has a direct relation to the overall performance of the system. This chapter will focus on the impact the memory subsystem has on the overall performance of the Pentium processor. First, the simulated performance of the DRAM subsystem for the Pentium processor covered in the design example portion of this application note will be presented. This will be followed by examining the impact of varying different parameters in the design example and how they affect the overall performance of the Pentium processor. To examine these effects, the traces captured from the applications described in Section 2.0 were fed through the performance simulator and configurations varied to measure the performance under different conditions.

3.1 Overall Performance

Figure 2 shows the simulated performance of the Pentium processor using the memory subsystem presented in the design example. The performance is characterized for five different applications and compared relative to an ideal zero wait state memory system. The performance level of the ideal memory system (i.e. with an infinite L1 cache) is the highest achievable and therefore normalized to 100%. Table 3 provides the parameters that were used in simulating the performance of this system.

The simulated performance of this memory subsystem ranges from 80% to 94% relative to an ideal memory system. The relative performance is lowest for Excel running under Windows as it has the lowest hit rate in the L1 cache compared to the other applications. CLIP on the other hand achieves the highest hit rate in the L1 cache relative to the rest of the applications and therefore shows the best performance.

Table 3. Configuration Parameters for the Pentium™ Processor DRAM Subsystem Design Example

	Configuration Parameters
Memory Bus Width	64 Bits
Memory Bus Speed	Full Speed
Memory Bus Pipelined	Yes
DRAM Subsystem Interleaving Factor	2-Bank Interleaved
DRAM Subsystem Paged	Yes (8 KByte Page Size)
Timing Read Page Hit	5-2-2-2 (Non-Pipelined); 4-2-2-2 (Pipelined)
Timing Read Page Miss	11-2-2-2 (Non-Pipelined); 9-2-2-2 (Pipelined)
Timing Write Page Hit	3 (Non-Pipelined); 3 (Pipelined)
Timing Write Page Miss	7 (Non-Pipelined); 5 (Pipelined)
Timing Writeback Page Hit	3-2-2-2
Timing Writeback Page Miss	7-2-2-2

NOTE:

Timing is given in CPU clocks; 5-2-2-2 refers to 5 clocks for lead off-cycle and 2 clocks for each successive access of the burst cycle. For further details refer to the design example section.

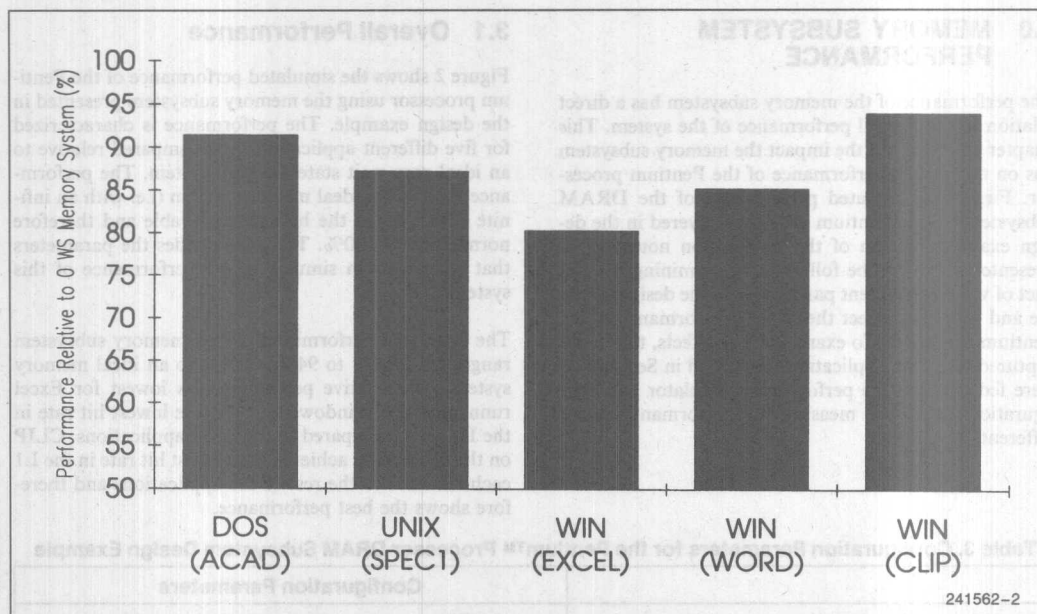


Figure 2. Overall Performance of the Pentium™ Processor Memory Subsystem Design Example

3.2 Impact of Wait States on Performance

This section examines the impact of wait states on overall performance of the Pentium processor. As wait states are added to the different read and write cycles, the number of clocks required to execute an application increases and the overall performance decreases. Three sets of simulations were run (one application from each operating system environment) in which the timing parameters from only one category were changed while the other parameters were kept constant. This was done so as to analyze the dependency of Pentium processor performance on each of the different timing parameters. The memory subsystem parameters shown in Table 3 were used as the base in the analysis.

3.2.1 READ CYCLE PERFORMANCE

The different memory timing parameters that are of interest in analyzing the read cycle performance are:

- Clock latency on read lead-off cycle (i.e., the number of clocks required to complete the first access of a burst read cycle or single read cycle).
- Clock latency on burst read cycles (i.e., the number of clocks required to complete the subsequent accesses of a burst read cycle).

Figure 3 shows the effects of burst read cycle timing on performance. It shows the difference in performance relative to a zero wait system as wait states are added. Only the burst read timing parameters are varied while the other parameters are kept constant as in Table 3.

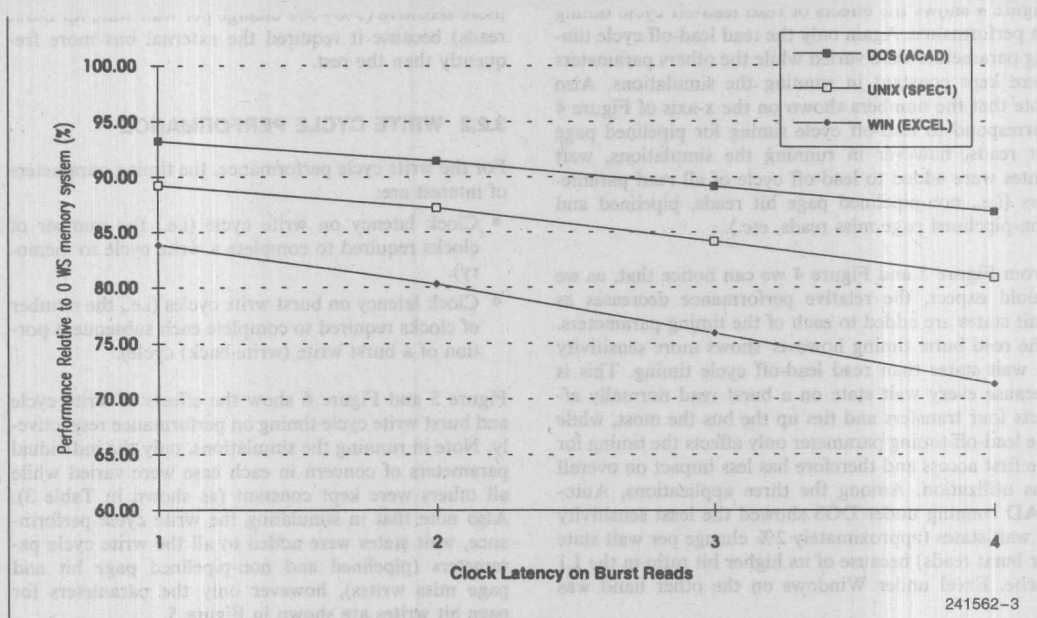


Figure 3. Effects of Burst Read Cycle Timing on Performance

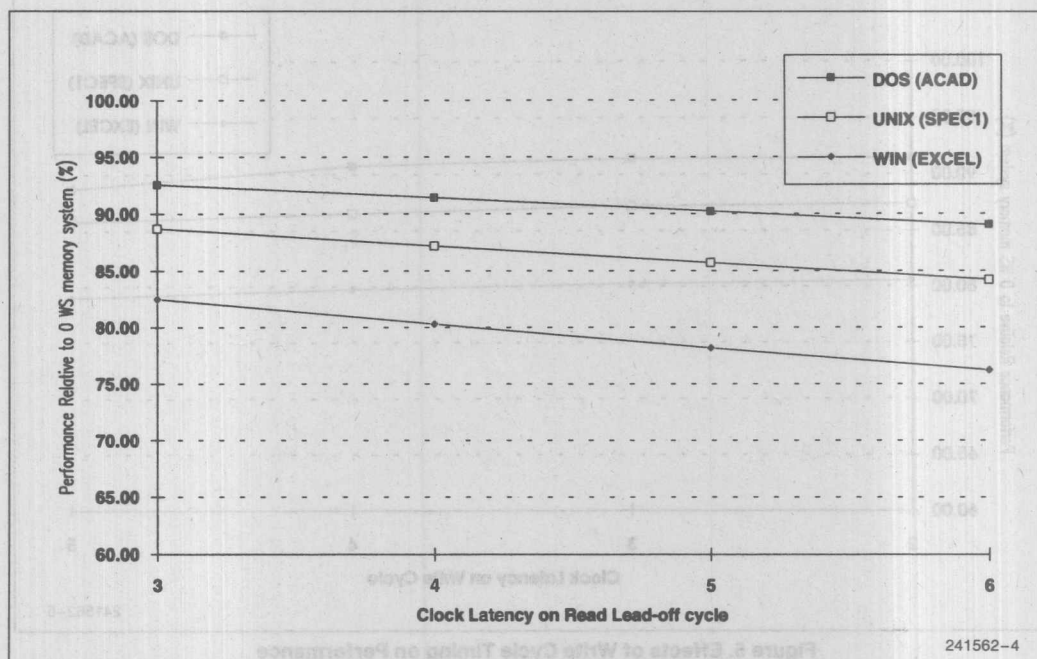


Figure 4. Effects of Read Lead-Off Cycle Timing on Performance

Figure 4 shows the effects of read lead-off cycle timing on performance. Again only the read lead-off cycle timing parameters were varied while the others parameters were kept constant in running the simulations. Also note that the numbers shown on the x-axis of Figure 4 correspond to lead-off cycle timing for pipelined page hit reads, however in running the simulations, wait states were added to lead-off cycle of all read parameters (i.e., non-pipelined page hit reads, pipelined and non-pipelined page miss reads, etc.).

From Figure 3 and Figure 4 we can notice that, as we would expect, the relative performance decreases as wait states are added to each of the timing parameters. The read burst timing however shows more sensitivity to wait states than read lead-off cycle timing. This is because every wait state on a burst read normally affects four transfers and ties up the bus the most, while the lead-off timing parameter only affects the timing for the first access and therefore has less impact on overall bus utilization. Among the three applications, AutoCAD running under DOS showed the least sensitivity to wait states (approximately 2% change per wait state for burst reads) because of its higher hit ratio in the L1 cache. Excel under Windows on the other hand was

more sensitive (3%–5% change per wait state for burst reads) because it required the external bus more frequently than the rest.

3.2.2 WRITE CYCLE PERFORMANCE

For the write cycle performance, the timing parameters of interest are:

- Clock latency on write cycle (i.e., the number of clocks required to complete a write cycle to memory).
- Clock latency on burst write cycles (i.e., the number of clocks required to complete each subsequent portion of a burst write (write-back) cycle).

Figure 5 and Figure 6 show the effects of write cycle and burst write cycle timing on performance respectively. Note in running the simulations, only the individual parameters of concern in each case were varied while all others were kept constant (as shown in Table 3). Also note that in simulating the write cycle performance, wait states were added to all the write cycle parameters (pipelined and non-pipelined page hit and page miss writes), however only the parameters for page hit writes are shown in Figure 5.

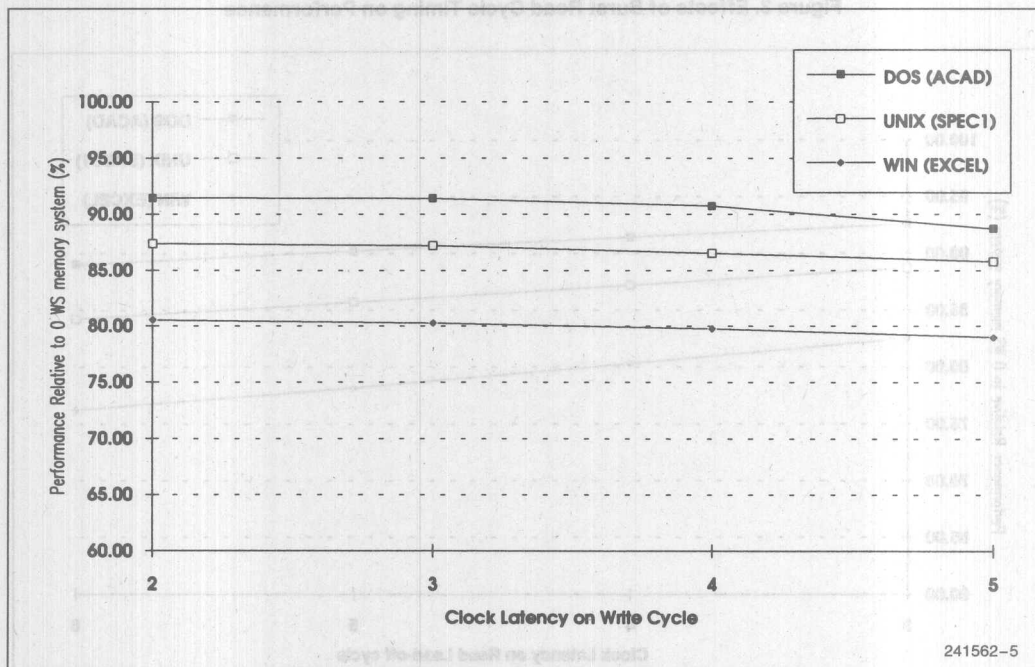


Figure 5. Effects of Write Cycle Timing on Performance

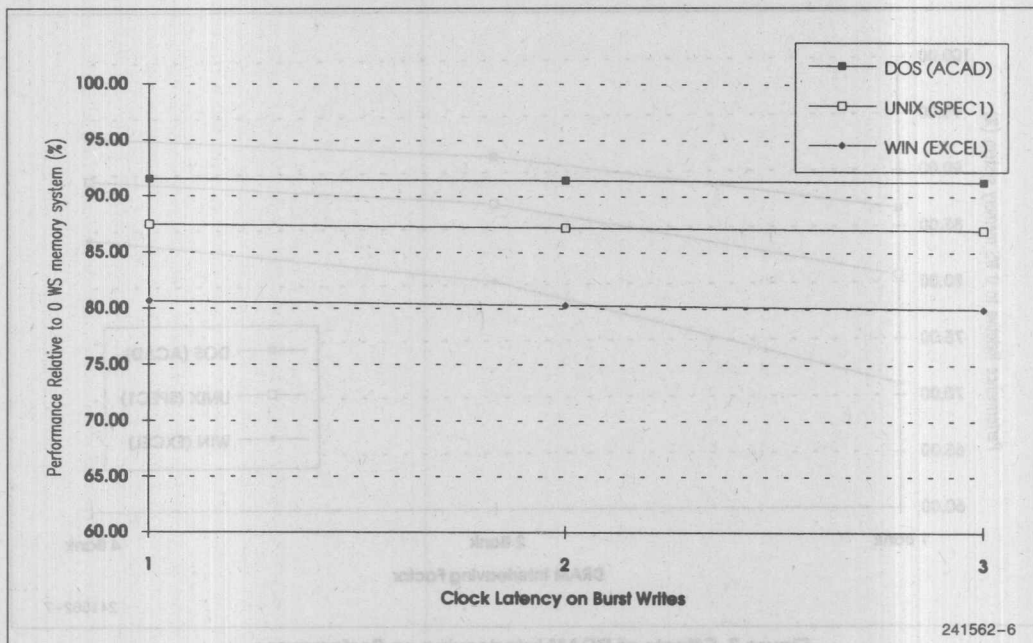


Figure 6. Effects of Burst Write Cycle Timing on Performance

Compared to read cycles, write cycles do not show as much sensitivity to wait states. This is mainly attributed to the write-back cache architecture of the Pentium processor's internal data cache which reduces the number of write cycles coming out on the external bus. It is also related to the fact that write cycles get posted to the write buffers internal to the Pentium processor, and the CPU does not get stalled unless there is another miss to L1 before the write buffer is emptied.

For burst write cycles, an additional wait state hardly impacts overall performance. This is attributed to the fact that write bursts (due to a read miss in the L1 cache) only constitute a small percentage of overall bus activity (see Figure 1) and never stall the CPU engine. In systems which involve heavy snooping activity, the number of write bursts may constitute a larger percentage of overall bus activity and therefore show a more significant impact to overall performance.

3.3 Effects of DRAM Interleaving on Performance

Interleaving involves using more than one bank of DRAM to effectively increase overall performance, with each bank normally having the same width as the memory bus. Each bank in the interleaved system is controlled separately. Thus, during an access to one bank, the other bank is made ready for the next sequential access. In this manner, data transfers can be carried out from alternate banks to achieve a faster burst transfer rate and increase the overall performance. Adding additional banks to the DRAM array however adds to the memory logic and minimum memory requirements and therefore involves a cost-performance tradeoff.

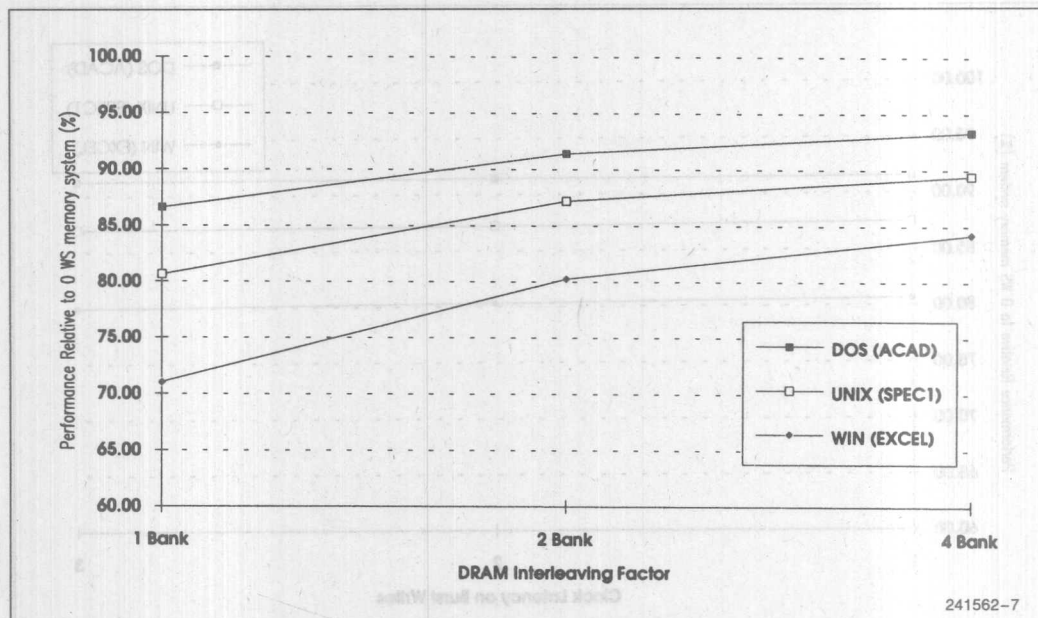


Figure 7. Effects of DRAM Interleaving on Performance

Figure 7 shows the effects of the DRAM interleaving factor (single bank versus a 2 bank and a 4 bank implementation) on the overall performance of the Pentium processor. The bus timing assumptions for the different interleaved systems are as follows:

- 1 Bank— 4 clock burst read timing, 3 clock burst write timing.
- 2 Bank— 2 clock burst read timing, 2 clock burst write timing.
- 4 Bank— 1 clock burst read timing, 1 clock burst write timing.

All other parameters remain constant as shown in Table 3.

The overall performance is compared relative to a zero wait state memory system. The results show that the single bank suffers as much as 9% loss in relative performance compared to a 2 bank system. While the 4 bank system adds 2%–4% to the overall performance for different applications over the 2 bank system. Since 4 bank interleaving adds cost to the memory subsystem by requiring additional logic and increasing the minimum DRAM size, the 2 bank interleaved system is therefore the best cost/performance option for the Pentium processor desktop.

3.4 Effects of DRAM Paging on Performance

DRAM paging provides faster accesses for successive data transfers that occur within the same page. It is typically used for all type of accesses (read, write, write-backs) in order to increase the overall performance. The assumption here is that the Row Address Strobe (RAS) signal remains asserted continuously and is deasserted only when there is a page miss.

Figure 8 shows the effects of DRAM paging on the overall performance of the Pentium processor. In this figure, a paged DRAM system with an 8K byte and a 16K byte page size is compared to a non-paged DRAM system and the performance comparison is relative to a zero wait state memory system. The timing assumptions for the non-paged memory system are as following:

Burst Read Cycle = 8-2-2-2;
 Burst Write Cycle = 6-2-2-2;
 Write Cycle = 6 clocks

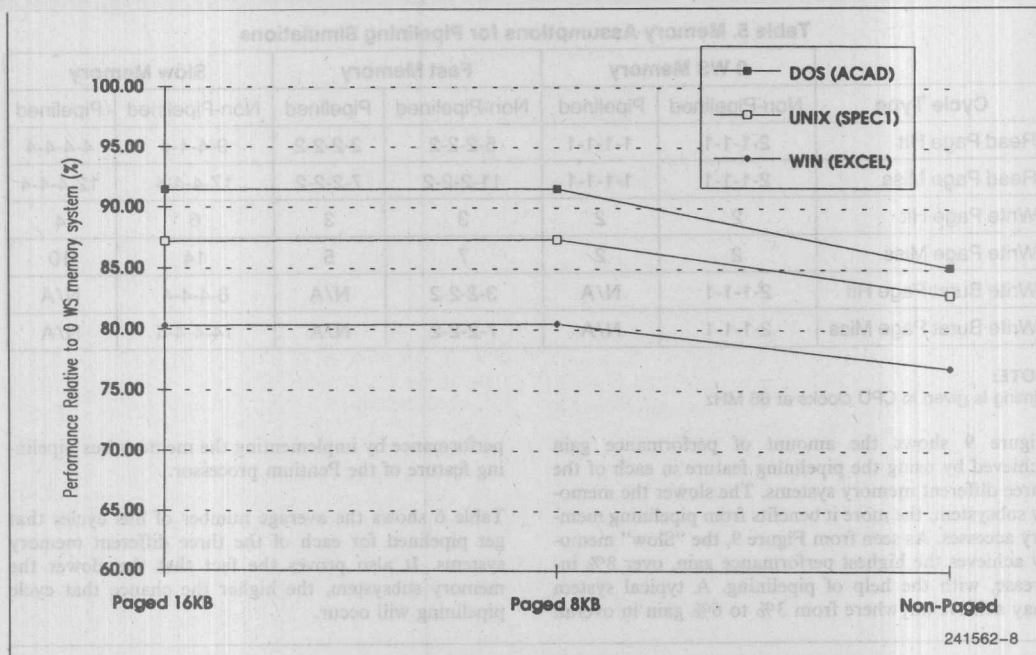


Figure 8. Effects of DRAM Paging on Performance

Table 4 shows the average page hit/miss ratios for each of the three applications in Figure 8. The results indicate that a non-paged system may suffer anywhere from 3% to 7% loss in performance compared to a paged system. This is due to the high page hit ratio for each of the applications that were simulated, as shown in Table 4 (shown for 8 Kbyte page size). However increasing the page size from 8 Kbyte to 16 Kbyte had no significant impact on overall performance.

another cycle before the current one is completed. This provides faster timing for back to back accesses and results in increased overall performance.

The simulations to understand the performance gain achieved from using this feature were carried out for three different types of memory subsystems. The cycle timings for these memory subsystems are detailed in Table 5.

3.5 Effects of Memory Bus Pipelining

The Pentium processor supports a bus pipelining feature for read and write cycles which allows it to drive

Table 4. Average Page Hit/Miss Ratio for Three Applications

	DOS (AutoCAD)	UNIX (Spec1)	Windows (Excel)
Read Page Hit Ratio	63.6%	56.6%	52.5%
Write Page Hit Ratio	91.2%	72.2%	52.5%
Prefetch Page Hit Ratio	66.6%	62.1%	66.0%

Table 5. Memory Assumptions for Pipelining Simulations

Cycle Type	0 WS Memory		Fast Memory		Slow Memory	
	Non-Pipelined	Pipelined	Non-Pipelined	Pipelined	Non-Pipelined	Pipelined
Read Page Hit	2-1-1-1	1-1-1-1	5-2-2-2	2-2-2-2	9-4-4-4	4-4-4-4
Read Page Miss	2-1-1-1	1-1-1-1	11-2-2-2	7-2-2-2	17-4-4-4	12-4-4-4
Write Page Hit	2	2	3	3	6	4
Write Page Miss	2	2	7	5	14	10
Write Burst Page Hit	2-1-1-1	N/A	3-2-2-2	N/A	6-4-4-4	N/A
Write Burst Page Miss	2-1-1-1	N/A	7-2-2-2	N/A	14-4-4-4	N/A

NOTE:

Timing is given in CPU clocks at 66 MHz

Figure 9 shows the amount of performance gain achieved by using the pipelining feature in each of the three different memory systems. The slower the memory subsystem, the more it benefits from pipelining memory accesses. As seen from Figure 9, the "Slow" memory achieves the highest performance gain, over 8% increase, with the help of pipelining. A typical system may achieve anywhere from 3% to 6% gain in overall

performance by implementing the memory bus pipelining feature of the Pentium processor.

Table 6 shows the average number of bus cycles that get pipelined for each of the three different memory systems. It also proves the fact that the slower the memory subsystem, the higher the chance that cycle pipelining will occur.

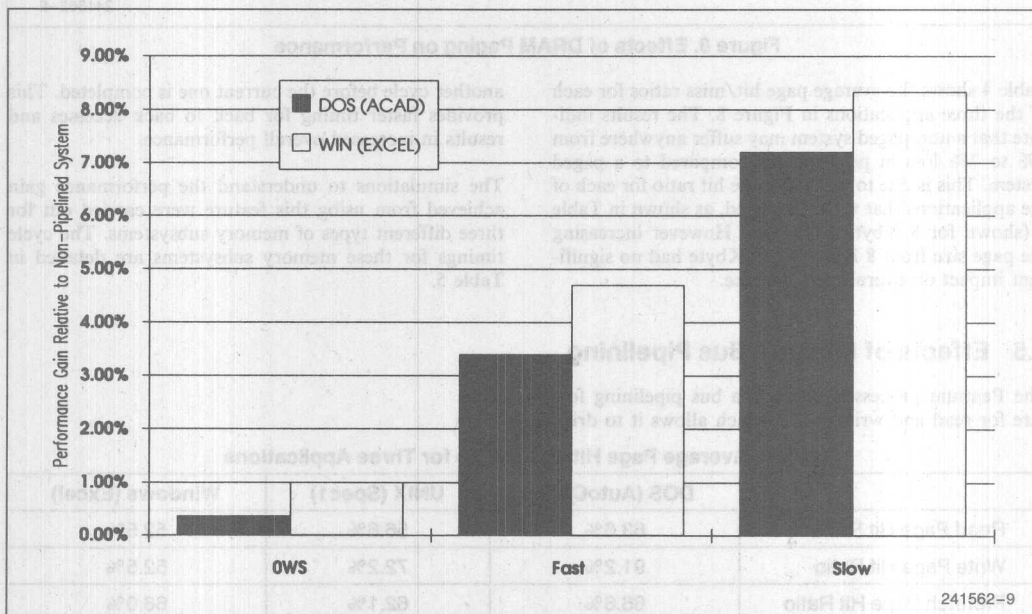


Figure 9. Effects of Memory Bus Pipelining on Performance

Table 6. Average Pipelining Ratio for Different Memories

	0 WS Memory		Fast Memory		Slow Memory	
	DOS (AutoCAD)	Windows (Excel)	DOS (AutoCAD)	Windows (Excel)	DOS (AutoCAD)	Windows (Excel)
Pipelined Read Ratio	1.6%	4.2%	28%	26%	47%	49%
Pipelined Write Ratio	0.8%	2.4%	5.2%	12%	85%	45%

3.6 Memory Bus Width and Bus Speed

The memory bus width and bus speed also affect the memory latency time and can impact overall performance. The Pentium processor is designed with a 64-bit data bus. Therefore it expects all data transfers to occur in 64 bits. Similarly the bus is designed to support a frequency of 66 MHz. Reducing the bus speed to 33 MHz would require extra clocks to complete a bus cycle and will increase the bus utilization, thereby reducing the overall performance.

Figure 10 shows the impact to overall performance in going from a 64-bit bus to a 32-bit bus, while Figure 11 shows the impact to overall performance in reducing the bus frequency from 66 MHz to 33 MHz. The assumptions for the timing of the memory systems for 32-bit bus and 33-MHz bus speed are given in Table 7. For comparison with the normal case (i.e., full speed bus, 64-bit data bus width), the parameters provided earlier in Table 3 were used.

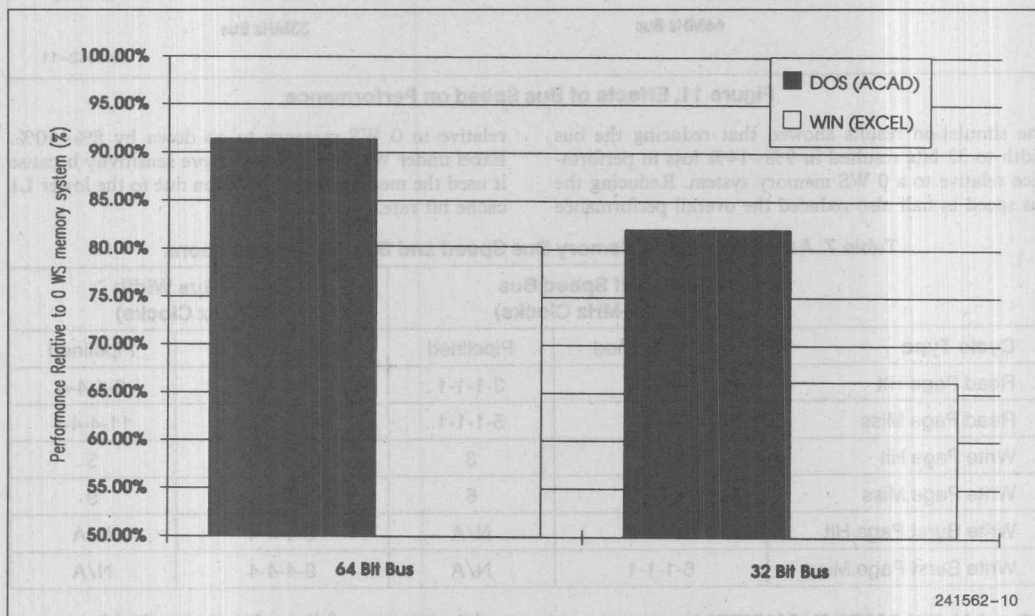


Figure 10. Effects of Bus Width on Performance

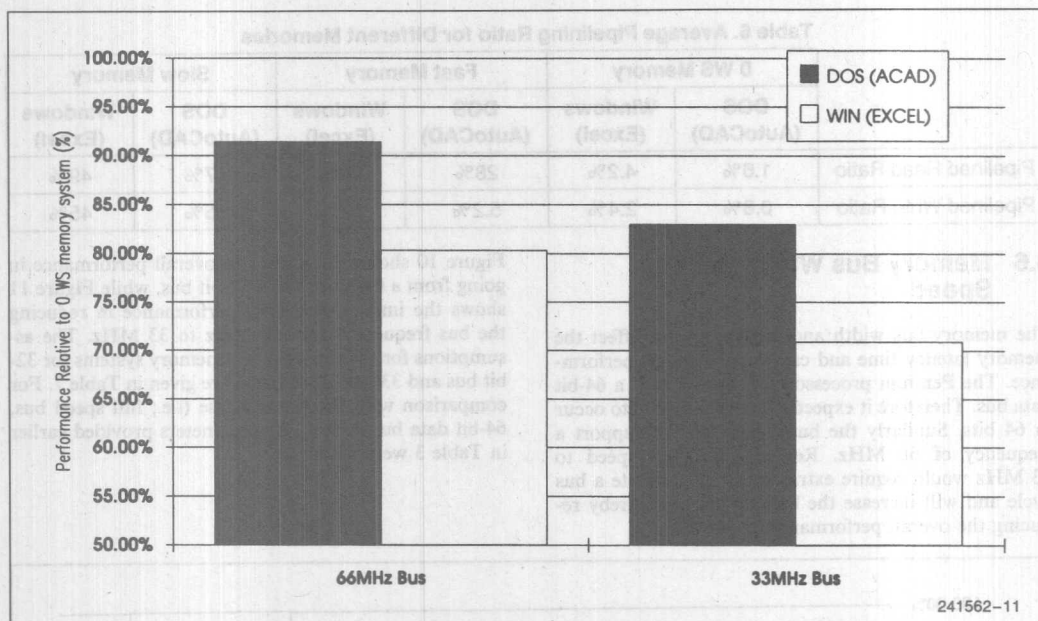


Figure 11. Effects of Bus Speed on Performance

The simulation results showed that reducing the bus width to 32 bits resulted in 9%–14% loss in performance relative to a 0 WS memory system. Reducing the bus speed in half also reduced the overall performance

relative to 0 WS memory to go down by 8%–10%. Excel under Windows showed more sensitivity because it used the memory bus more often due to the lower L1 cache hit rate.

Table 7. Assumptions for Memory Bus Speed and Bus Width Simulations

Cycle Type	Half Speed Bus (33-MHz Clocks)		32-Bit Bus Width (66-MHz Clocks)	
	Non-Pipelined	Pipelined	Non-Pipelined	Pipelined
Read Page Hit	4-1-1-1	3-1-1-1	7-4-4-4	6-4-4-4
Read Page Miss	7-1-1-1	5-1-1-1	13-4-4-4	11-4-4-4
Write Page Hit	3	3	5	5
Write Page Miss	6	6	9	9
Write Burst Page Hit	3-1-1-1	N/A	5-4-4-4	N/A
Write Burst Page Miss	6-1-1-1	N/A	9-4-4-4	N/A

4.0 AN EXAMPLE SYSTEM ARCHITECTURE OVERVIEW

The next section will work with an example to discuss memory subsystem design for the Pentium processor. The block diagram for an example system architecture is shown in Figure 12. The assumptions for the system

architecture are as follows: It is designed without a second level cache, with the memory subsystem residing on the CPU (memory) bus, running at full speed, with a 64-bit data path to the Pentium processor. The memory subsystem portion consists of a two bank interleaved DRAM array that supports DRAM paging, memory cycle pipelining and CPU write-back operation. In or-

der to provide interface to an existing system chip set, the bus conversion logic provides the necessary logic required to interface to the Intel 82350 EISA chip set as shown in the block diagram. The discussion in the following sections will focus on detailing the memory subsystem and the bus conversion logic that is required to interface to a 32-bit, 33-MHz bus.

5.0 DRAM SUBSYSTEM OVERVIEW

This section presents a functional overview of the example memory subsystem. A block diagram of the memory subsystem is shown in Figure 13. The subsystem has been designed to support as many functions of the CPU as possible, such as, allowing CPU to operate in write-back mode; supporting bus pipelining; support of burst read and burst write cycles, etc. The other attributes of the memory subsystem include DRAM paging support and two bank interleaving.

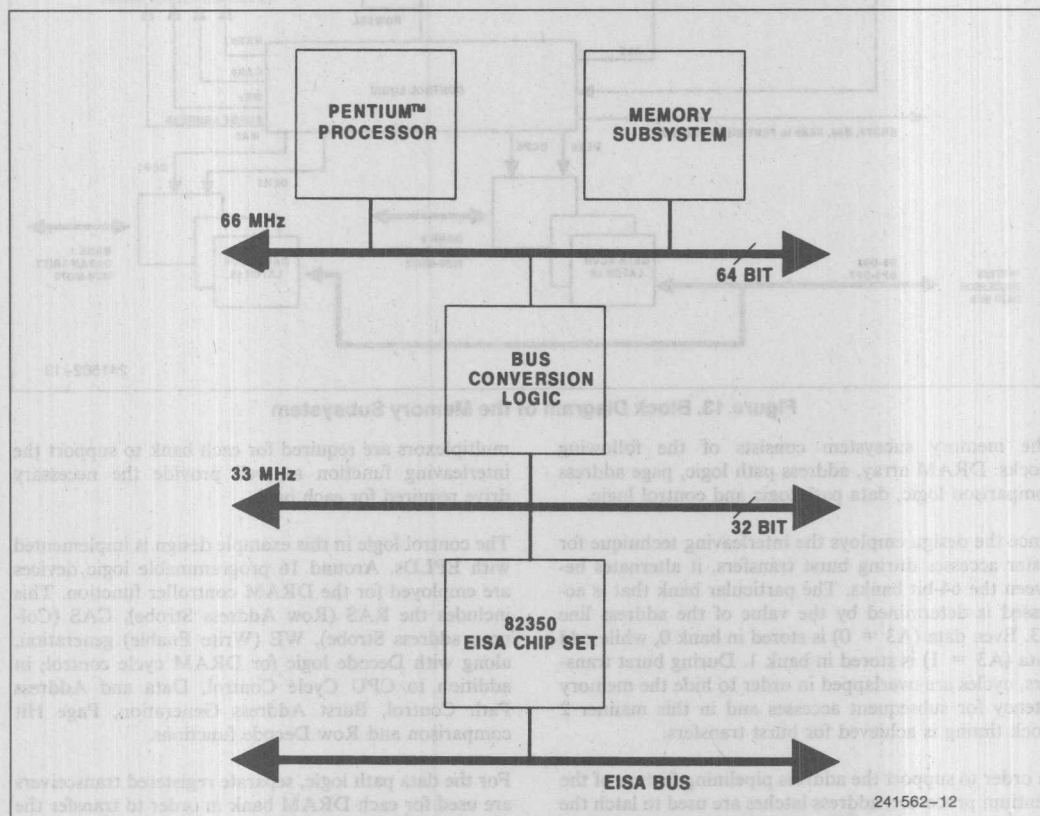


Figure 12. System Architecture Overview

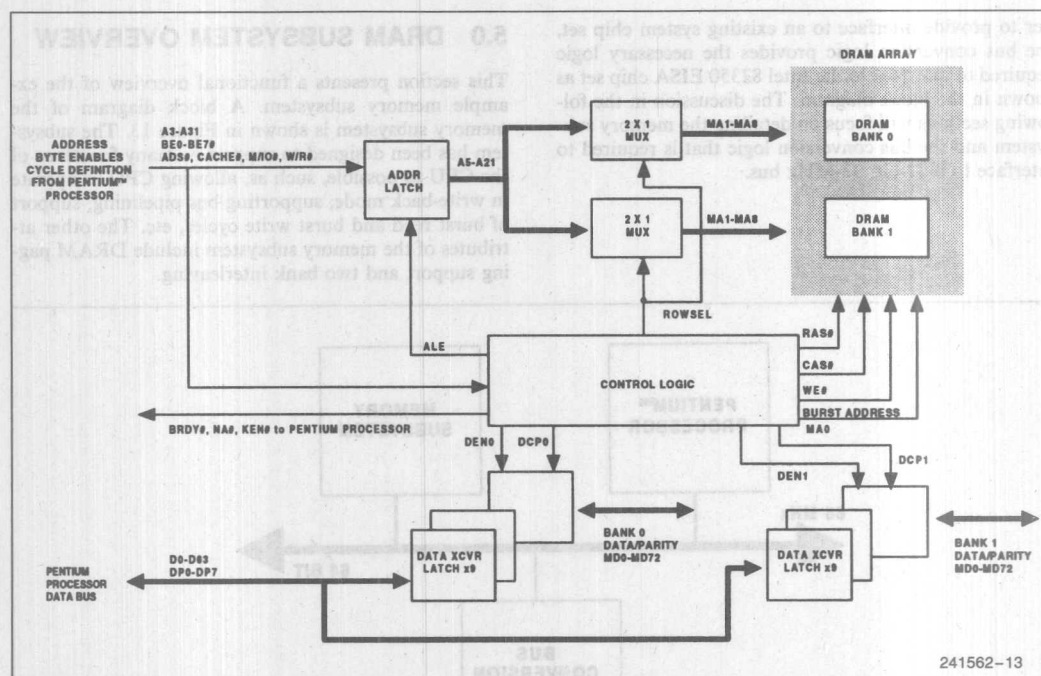


Figure 13. Block Diagram of the Memory Subsystem

The memory subsystem consists of the following blocks: DRAM array, address path logic, page address comparison logic, data path logic and control logic.

Since the design employs the interleaving technique for faster accesses during burst transfers, it alternates between the 64-bit banks. The particular bank that is accessed is determined by the value of the address line A3. Even data (A3 = 0) is stored in bank 0, while odd data (A3 = 1) is stored in bank 1. During burst transfers, cycles are overlapped in order to hide the memory latency for subsequent accesses and in this manner 2 clock timing is achieved for burst transfers.

In order to support the address pipelining feature of the Pentium processor, address latches are used to latch the Pentium processor address as soon as a bus cycle is started. Once the address is latched, the control logic asserts the NA# signal to allow the Pentium processor to issue another cycle before the previous cycle gets completed. This effectively reduces the lead-off cycle time by 2 to 3 clocks for back to back transfers and improves overall performance.

Multiplexor logic is used to drive the appropriate row and column addresses to the DRAMs. Separate address

multiplexors are required for each bank to support the interleaving function as well provide the necessary drive required for each bank.

The control logic in this example design is implemented with EPLDs. Around 16 programmable logic devices are employed for the DRAM controller function. This includes the RAS (Row Address Strobe), CAS (Column address Strobe), WE (Write Enable) generation, along with Decode logic for DRAM cycle control; in addition to CPU Cycle Control, Data and Address Path Control, Burst Address Generation, Page Hit comparison and Row Decode functions.

For the data path logic, separate registered transceivers are used for each DRAM bank in order to transfer the data between the CPU and the DRAMs. These data transceivers are enabled/disabled appropriately to switch between bank 0 and bank 1 during bursting. The registered function of these transceivers is used for supporting the pipeline operation for write cycles. Write data coming out of the CPU is latched in these registers and then written to DRAMs while the CPU is allowed to issue another cycle. In this manner, write cycles can complete in three clocks and incur additional wait states only if there is a page miss.

Details on each of the blocks that make up the DRAM subsystem will be covered in subsequent sections.

5.1 DRAM Organization

The DRAM array is organized in two banks of 64-bit, interleaved memory. Each bank is two rows deep (Rows A and B). Figure 14 shows how the DRAM array is organized.

Each of the two banks require 60 ns Fast Page Mode DRAMs in single density 256K x 36 or double density 512K x 36 memory module configurations. The minimum memory configuration for this memory subsystem

is 4 Mbytes using four 256K x 36 memory modules (2 modules per bank); and the maximum is 16 Mbytes using eight 512K x 36 memory modules (4 modules per bank). With a few restrictions a combination of single and double density modules is also permitted.

The DRAM control signals, RAS#, CAS#, WE#, and address lines are distributed according to the particular bank, the row within the bank, and the loading on these signals. There are a total of 16 CAS# signals, 8 RAS# signals and 4 WE# signals generated to cover the entire DRAM array. The address lines are distributed such that each single line covers only one row per bank.

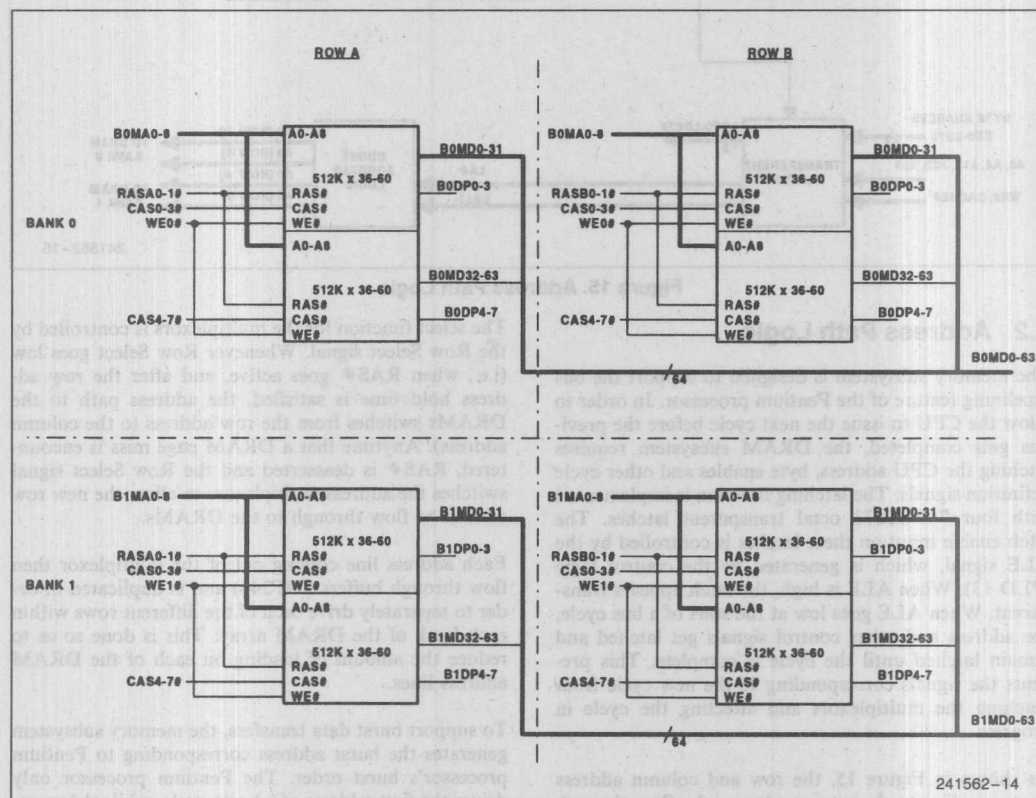


Figure 14. Organization of the DRAM Array

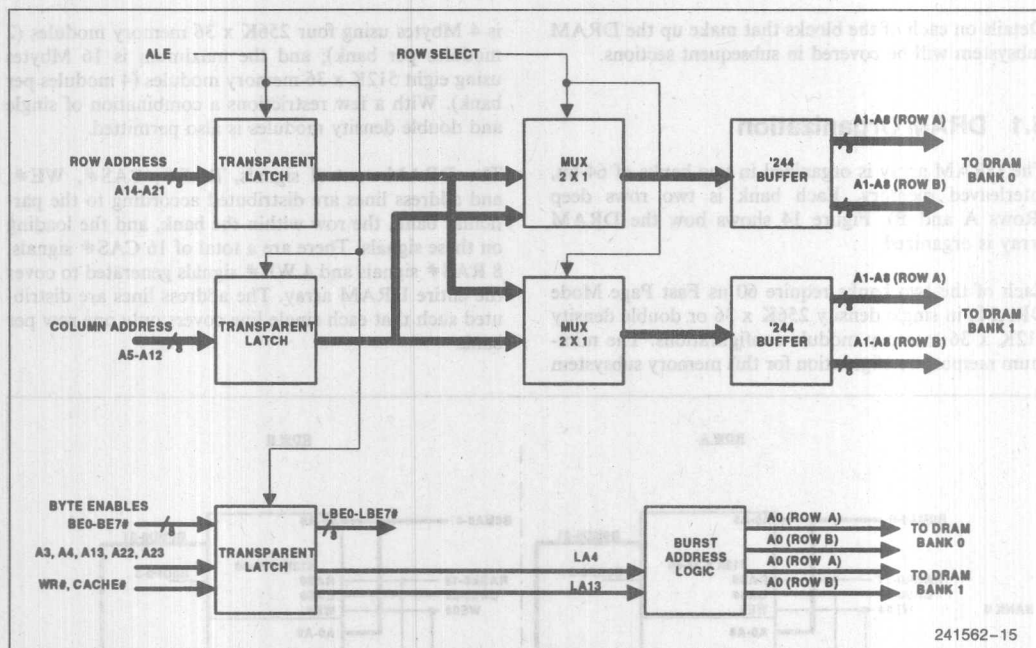


Figure 15. Address Path Logic

5.2 Address Path Logic

The memory subsystem is designed to support the bus pipelining feature of the Pentium processor. In order to allow the CPU to issue the next cycle before the previous gets completed, the DRAM subsystem requires latching the CPU address, byte enables and other cycle definition signals. The latching function is implemented with four 74FCT573 octal transparent latches. The latch enable input on these latches is controlled by the ALE signal, which is generated by the control logic (PLD 13). When ALE is high, the latch appears transparent. When ALE goes low at the start of a bus cycle, the address and other control signals get latched and remain latched until the cycle is complete. This prevents the signals corresponding to the new cycle from reaching the multiplexors and affecting the cycle in progress.

As shown in Figure 15, the row and column address corresponding to the memory bus cycle, flow through the transparent latches and reach the address multiplexors for each bank. Four 74F257 quad 2 to 1 multiplexors are used for driving the appropriate row and column addresses to the DRAMs. The multiplexors are separate for each bank, which is required for the interleaving function.

The select function for the multiplexors is controlled by the Row Select signal. Whenever Row Select goes low (i.e., when RAS# goes active, and after the row address hold time is satisfied, the address path to the DRAMs switches from the row address to the column address). Anytime that a DRAM page miss is encountered, RAS# is deasserted and the Row Select signal switches the address multiplexors to allow the new row address to flow through to the DRAMs.

Each address line coming out of the multiplexor then flow through buffers (74F244) and is duplicated in order to separately drive each of the different rows within each bank of the DRAM array. This is done so as to reduce the amount of loading on each of the DRAM address lines.

To support burst data transfers, the memory subsystem generates the burst address corresponding to Pentium processor's burst order. The Pentium processor only drives the first address of a burst cycle, while the memory subsystem generates the addresses for the remaining transfers of the burst. In this design, the control logic (PLD 14) generates the burst address and drives

implements the multiplexing function for address line A0 to the DRAMs such that A0 corresponds to CPU

when the row address flows through to the DRAMs.

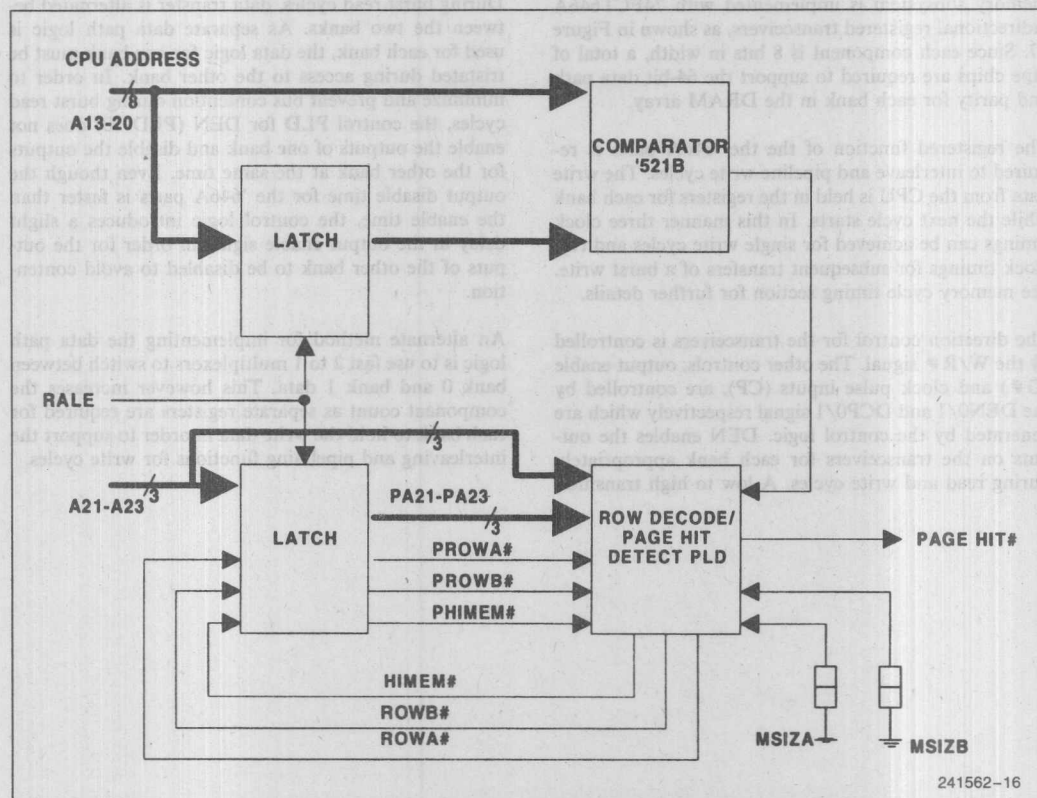


Figure 16. DRAM Page Address Comparison Logic

Figure 16 shows the DRAM page address comparison logic. This logic compares the address of the current cycle with that of the previous cycle and generates a page hit signal. If the current cycle's address is to the same DRAM row (page) as the previous cycle, then the page hit signal is activated. In case of a page miss, the control logic, in response to the page hit signal being inactive, deasserts RAS# to satisfy the RAS precharge time and also supplies a new row address to the DRAMs. For back to back page hit cycles, RAS# remains active and thereby avoids the min. RAS# to CAS# active delay for each cycle.

An 8-bit identity comparator, 74FCT521B, is used for comparing the address lines A13-A20. The address lines A21-A23 are compared using a PLD. This PLD also checks whether the current row matches the previ-

ous row and bank and finally combines all the separate compare output signals to generate the DRAM page hit signal.

The page hit detect PLD (PLD 9) also performs the row decode function. The jumpers MSIZA and MSIZB, as shown in Figure 16, set the appropriate memory size depending on how the DRAM banks are populated. MSIZA corresponds to row A, while MSIZB corresponds to row B. If MSIZA is set to zero, then row A is set for 256K x 36 bit modules, while MSIZA equal to 1 sets row A in each bank for 512K x 36-bit modules. The information provided by these jumper settings along with the status of address lines A22 and A23 determine the particular row that is accessed and is subsequently used by the RAS decode logic to strobe the appropriate row.

5.3 Data Path Logic

The data path logic (including data parity) for the memory subsystem is implemented with 74FCT646A bidirectional registered transceivers, as shown in Figure 17. Since each component is 8 bits in width, a total of nine chips are required to support the 64-bit data path and parity for each bank in the DRAM array.

The registered function of the the '646A parts is required to interleave and pipeline write cycles. The write data from the CPU is held in the registers for each bank while the next cycle starts. In this manner three clock timings can be achieved for single write cycles and two clock timings for subsequent transfers of a burst write. See memory cycle timing section for further details.

The direction control for the transceivers is controlled by the W/R# signal. The other controls, output enable (G#) and clock pulse inputs (CP), are controlled by the DEN0/1 and DCP0/1 signal respectively which are generated by the control logic. DEN enables the outputs on the transceivers for each bank appropriately during read and write cycles. A low to high transition

on the DCP signal allows the write data to be stored in the registers during write cycles.

During burst read cycles, data transfer is alternated between the two banks. As separate data path logic is used for each bank, the data logic for one bank must be tristated during access to the other bank. In order to minimize and prevent bus contention during burst read cycles, the control PLD for DEN (PLD 10) does not enable the outputs of one bank and disable the outputs for the other bank at the same time. Even though the output disable time for the '646A parts is faster than the enable time, the control logic introduces a slight delay in the output enable signal in order for the outputs of the other bank to be disabled to avoid contention.

An alternate method for implementing the data path logic is to use fast 2 to 1 multiplexors to switch between bank 0 and bank 1 data. This however increases the component count as separate registers are required for each bank to hold the write data in order to support the interleaving and pipelining functions for write cycles.

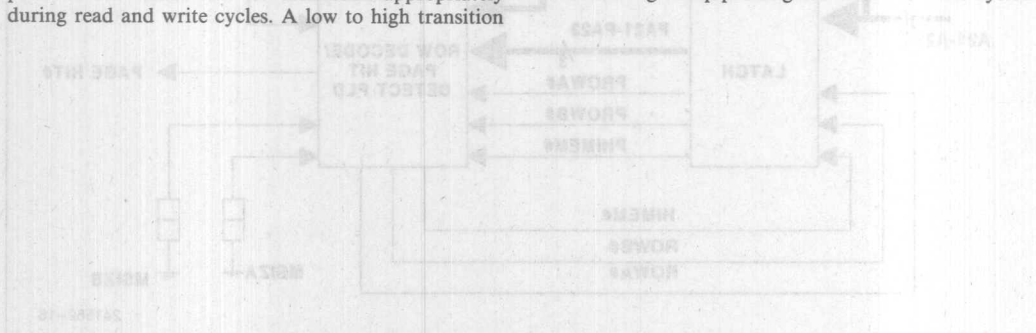


Figure 16. DRAM Page Address Comparison Logic

one row and bank and finally compares all the separate compare output signals to generate the DRAM page hit signal.

The page hit detect PLD (PLD 9) also performs the row decode function. The jumpers M21A and M21B, as shown in Figure 16, set the appropriate memory size depending on how the DRAM banks are populated. M21A corresponds to row A; while M21B corresponds to row B. If M21A is set to zero, then row A is set for 256K x 16 bit modules, while M21A equal to 1 sets row A to each bank for 512K x 16 bit modules. The information provided by these jumper settings along with the status of address lines A22 and A23 determine the particular row that is activated and is subsequently used by the RAS decoder logic to enable the appropriate row.

Figure 16 shows the DRAM page address comparison logic. This logic compares the address of the current cycle with the address of the previous cycle and generates a page hit signal. If the current cycle's address is to the same DRAM row (page) as the previous cycle, then the page hit signal is activated. In case of a page miss, the control logic in response to the page hit signal (page hit) generates a RAS# to activate the RAS# decoder, which then activates a new row address to the DRAM. For back to back page hit cycles, RAS# remains active and thereby avoids the min. RAS# to CAS# delay for each cycle.

An 8-bit 3-state comparator 74FCT221B is used for comparing the address lines A13-A10. The address lines A22-A23 are compared using a PLD. The PLD also checks to see if the current row matches the previ-

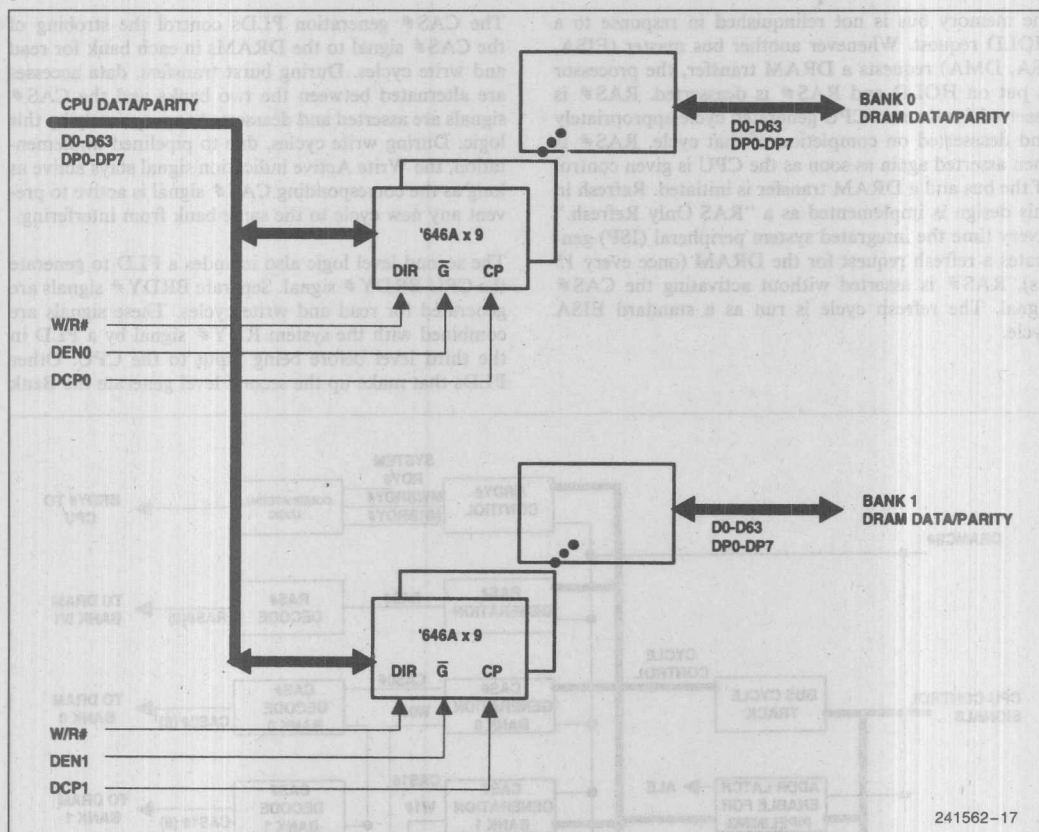


Figure 17. Data Path Logic

5.4 Control Logic

In this design, the logic responsible for generating all the signals necessary to control the memory subsystem is implemented in PLDs. Figure 18 shows how the control logic is distributed.

The first level of logic is made up of the memory bus cycle control for the CPU. The first level samples the cycle definition signals from the CPU and generates the appropriate signals to the second level of logic for DRAM cycle control. The second level therefore waits for the cycle start signal from the first level before initiating a DRAM cycle. Distributing the logic in this manner reduces the amount of loading on the CPU cycle definition signals, especially ADS#. It also means that the decode outputs (such as DRAM Chip Select signal and Page Hit signal) have an extra clock before they are sampled by the second level PLDs, therefore

more decode time is allowed. The first level logic is also responsible for generating the address latch enable signal and NA# signal back to the CPU for cycle pipelining.

The second level of logic generates the master signals such as RAS# and CAS# required for the DRAM access, in addition to signals that control the data transfer between the DRAM subsystem and the memory bus. As shown in Figure 18, one PLD generates the RAS# signal while one PLD each is dedicated to generating the CAS# and Write Active indication signals for each bank.

The RAS# Generation PLD works in conjunction with the RAS# decode PLD, which also implements the RAS# Precharge Count function, to keep RAS# continuously active for all CPU cycles as long as a page miss is not encountered, a RESET does not occur, or

the memory bus is not relinquished in response to a HOLD request. Whenever another bus master (EISA, ISA, DMA) requests a DRAM transfer, the processor is put on HOLD and RAS# is deasserted. RAS# is asserted for the non-CPU generated cycle appropriately and deasserted on completion of that cycle. RAS# is then asserted again as soon as the CPU is given control of the bus and a DRAM transfer is initiated. Refresh in this design is implemented as a "RAS Only Refresh." Every time the integrated system peripheral (ISP) generates a refresh request for the DRAM (once every 15 μ s), RAS# is asserted without activating the CAS# signal. The refresh cycle is run as a standard EISA cycle.

The CAS# generation PLDs control the strobing of the CAS# signal to the DRAMs in each bank for read and write cycles. During burst transfers, data accesses are alternated between the two banks and the CAS# signals are asserted and deasserted appropriately by this logic. During write cycles, due to pipelined implementation, the Write Active indication signal stays active as long as the corresponding CAS# signal is active to prevent any new cycle to the same bank from interfering.

The second level logic also includes a PLD to generate the CPU BRDY# signal. Separate BRDY# signals are generated for read and write cycles. These signals are combined with the system RDY# signal by a PLD in the third level before being input to the CPU. Other PLDs that make up the second-level generate the Bank

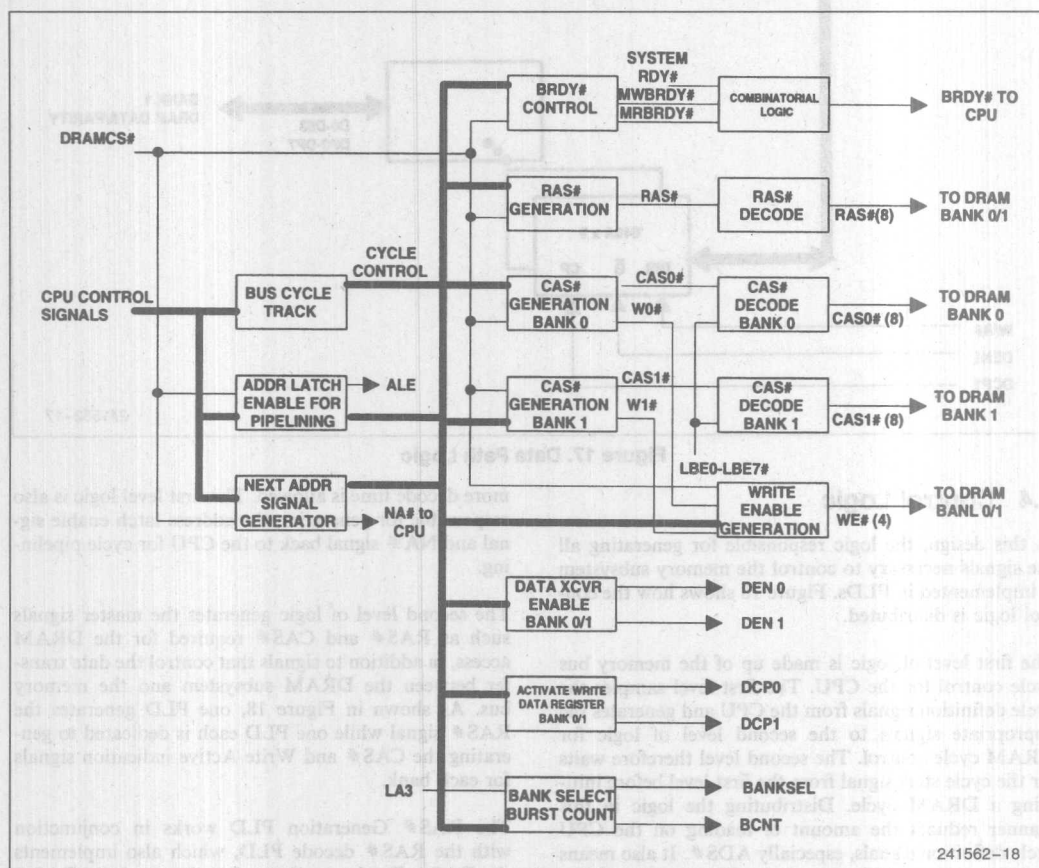


Figure 18. DRAM Subsystem Control Logic

Select and Burst Count Tracking signals along with the data tranceiver control signals for data path control.

The third level of logic is mainly combinatorial logic. This includes the Row Decode, RAS# Decode, CAS# decode and Write Enable Generation signals for each bank, and BRDY# combine logic. During write cycles, the CAS# decode logic qualifies the CAS# signal with the byte enable signals such that only those bytes for which the corresponding byte enable signals are assert- ed are written into.

For the Address Decode function, an 8K x 8 SRAM is used. This SRAM performs the DRAM Chip Select, Write Protect, and Cacheable Address Decode func- tions. Its implementation and programming is dis- cussed in reference [1] (*Intel486™ Microprocessor- Based 82350 EISA Design Guide*). The only difference in this design is that a faster SRAM chip is used, 10 ns instead of 15 ns, to allow the decode outputs to be available for sampling at the end of the second clock of the CPU bus cycle.

6.0 BUS CONVERSION LOGIC

In this example design, the memory subsystem resides on the CPU bus, running at full speed (66 MHz) with a 64-bit data path interface to the CPU. This section will examine the address/data bus conversion logic required

to interface this memory subsystem with the 82350 EISA chip set which is designed for a 33-MHz host bus interface and 32-bit data path. The address/data con- version logic is shown in Figure 19. It consists of Ad- dress Buffer logic, Data Bus Buffer logic, Data Parity Translate and Byte Enable Translate logic.

The address buffers are required to isolate the CPU address bus from the 82350 host address bus during CPU writeback cycles. Four 74F645 buffers are used for the 32-bit address bus. During EISA, ISA, DMA master activity on the memory bus, the CPU is snooped for every read or write cycle to DRAM. In case of a snoop hit to a modified line (which would require a writeback cycle), the HITM# signal goes active and the output buffers on the '645 chips are turned off to allow the CPU to drive the address for the writeback cycle on the memory bus. In all other cases, the outputs on these buffer chips stay turned on.

For the 64-bit to 32-bit data path interface, two EBB (EISA Bus Buffer) chips strapped in "data without parity" mode are used. This logic can alternatively be implemented with discrete logic using 74F543 and 74ALS245 chips. However the use of EBB chips reduces the overall chip count and board space as each EBB chip is able to provide a 32-bit interface.

3

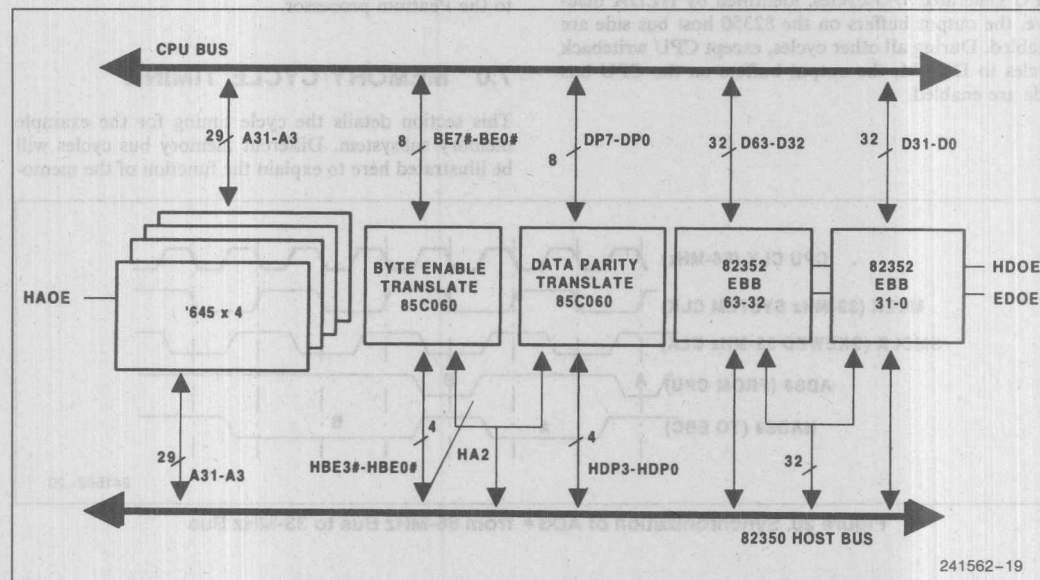


Figure 19. Address/Data Bus Conversion Logic

The controls for these data buffer chips is provided by the EDOE# and HDOE# signals. The HDOE# signal activates the output buffers on the CPU bus side during CPU I/O read cycles and write cycles generated by other bus masters (EISA/ISA/DMA) in the system. In case of a non-CPU generated write cycle to DRAM, the output enables on both high and low (Dwords) EBBs are enabled simultaneously and exactly the same data is duplicated at the outputs and appears on the CPU bus side. It is the byte enable signals that control which byte within the 64-bit word gets written into the DRAM array. For CPU generated I/O write cycles and EISA/ISA/DMA master generated read cycles to DRAM, the output buffers on the host bus side are enabled by the EDOE# signals. Depending on the status of the HA2 (Host Address line 2) signal, either the high 32-bits or the low 32-bits from the 64-bit bus appear on the 32-bit host data bus.

For the Data Parity and Byte Enable translation between the 64-bit bus and the 32-bit bus, two 85C060 PLDs are used. Each of these chips is designed with 16 I/O pins and therefore convenient to use for this purpose. The Byte Enable translation PLD converts the 8-byte Enables corresponding to a 64-bit bus appropriately into 4 Byte Enables corresponding to a 32-bit bus and vice versa. Similarly the Data Parity translate PLD does the same for the Data Parity signals. The output enables for the I/O pins on these chips is controlled by the HLDA and HITM# signals from the CPU. During CPU generated I/O cycles, identified by HLDA inactive, the output buffers on the 82350 host bus side are enabled. During all other cycles, except CPU writeback cycles to DRAM, the output buffers on the CPU bus side are enabled.

6.1 Synchronization of Control Signals

The synchronization of signals between the Pentium processor (66 MHz) and the EISA bus controller (33 MHz) are performed by two PLDs. One of the PLDs takes some of the CPU output signals as inputs and regenerates these for the 33-MHz bus. The other PLD synchronizes the input control signals going to the CPU from the 33-MHz bus.

Two of the Pentium processor's output signals, ADS#, and PCHK#, are available for only one clock coming out of the CPU at 66 MHz. These have to regenerate for the 33-MHz bus in order for the bus controller to sample them. Figure 20 shows how the ADS# signal is synchronized and regenerated for the 33-MHz bus. If the ADS# signal comes out relative to the falling edge of the 33-MHz clock (label A), the output HADS# gets sampled at the next rising edge of MCLK (i.e., next rising edge of CPU clock). However, if ADS# is output relative to the rising edge of MCLK (label B), the next rising edge of MCLK coinciding with rising edge of CPU clock happens two CPU clocks later as shown. The synchronization of the PCHK# signal is performed identical to the ADS# signal.

The other PLD handles the synchronization of CPU input signals like EADS#, HOLD, FLUSH# and BRDY# coming from the EBC before they are input to the Pentium processor.

7.0 MEMORY CYCLE TIMING

This section details the cycle timing for the example memory subsystem. Different memory bus cycles will be illustrated here to explain the function of the memo-

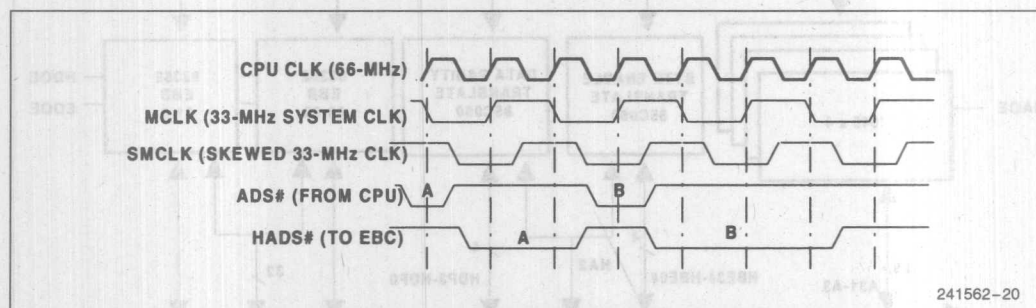


Figure 20. Synchronization of ADS# from 66-MHz Bus to 33-MHz Bus

ry control logic. Note that the bus cycles shown in this section are specific to this example and are not restricted to the timing provided here. Please refer to the *Pentium™ Processor Data Book*, for the description and timing of the CPU specific signals.

Table 8 details the clock timing for different type of read and write cycles that apply to this memory subsystem. These will be explained in subsequent sections. Note, the numbers shown indicate the actual number of CPU clocks that would be required to complete the bus cycles. For example 5-2-2-2 refers to five clocks for the first access and two clocks for each subsequent access of the burst.

7.1 Read Cycle Timing

7.1.1 NON-PIPELINED READ CYCLE AFTER RESET

Figure 21 shows a CPU-generated read cycle to DRAM that is started after a RESET or DRAM Refresh. In this type of cycle, the DRAM access is started in the third clock with the activation of the RAS# signal. Since this memory subsystem is designed to support paging, RAS# is normally kept active unless there is a page miss to the DRAMs (explained later). The actual number of clocks that are required to complete the

DRAM access from the assertion of the RAS# signal are dictated by the DRAM timings. In this design 60-ns DRAMs are used therefore 6 clocks are required to complete the first data access from RAS# for a total of 8 clocks to complete the first data access of the bus cycle (from ADS# to the first BRDY#).

For the read cycle, the memory controller functions as follows. The CPU cycle control logic samples the ADS# signal along with other cycle definition signals at the start of the second bus cycle and generates a DRAM cycle in progress signal. This signal along with the DRAM chip select signal from the address decode logic is input to the actual RAS# and CAS# signal generation PLDs to be sampled at the start of the third bus cycle. The page hit/miss signal is also sampled in this clock, but since this cycle started with RAS# inactive, the page hit signal is not factored in and the DRAM cycle is initiated by the assertion of RAS#.

The KEN# (cache enable) signal is available from the decode logic in the second clock. After it is determined that the bus cycle is a valid DRAM cycle, the KEN# signal is asserted to the CPU to start the cache line fill. This is done in the third clock since the KEN# signal is sampled by the CPU in the same clock in which the first BRDY# (or NA#) is asserted. The KEN# signal then stays active until the end of the cycle.

3

Table 8. Cycle Timing for the Example Memory Subsystem

	Pipelined	Non-Pipelined
Read Cycle		
Burst Read Page Hit following a Read Cycle	4-2-2-2	5-2-2-2
Burst Read Page Hit following a Write Cycle	6-2-2-2	5-2-2-2
Burst Read Page Miss following a Read Cycle	9-2-2-2	11-2-2-2
Burst Read Page Miss following a Write Cycle	11-2-2-2	11-2-2-2
Write Cycle		
Write Page Hit following a Read or Write Cycle	3	3
Write Page Miss following a Write Cycle	7	7
Write Page Miss following a Read Cycle	5	7
Burst Writeback Page Hit Cycle	N/A	3-2-2-2
Burst Writeback Page Miss Cycle	N/A	7-2-2-2

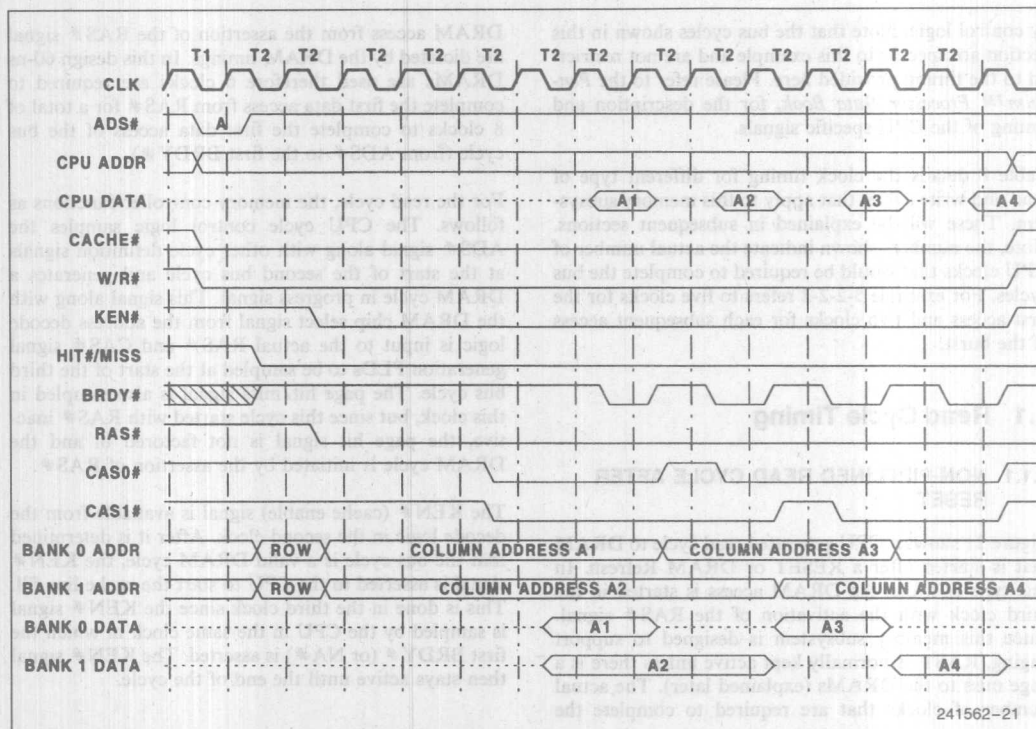


Figure 21. DRAM Read Cycle after RESET

The address logic makes the new row address available to the DRAM array for both banks in the first T2 of the bus cycle before activation of the RAS# signal. The Row Select signal then switches the address multiplexers for each bank to the column address path, after the row address hold time is satisfied.

Since interleaving is implemented in this design, the CAS# signals for both banks are asserted simultaneously in the sixth clock after satisfying the minimum RAS# to CAS# active delay. In the bus cycle shown, the first access is made from bank 0. The BRDY# signal to complete the first transfer is made three clocks after the CAS0# signal. The three clock timing is required to meet the CAS# to data active delay for the DRAMs as well as the propagation delay through the data transceivers. Once the first transfer is complete, the data path switches to bank 1 to complete the data access for the second burst transfer. The CAS# signal for bank 0 is precharged during this time to start

the third access. In this manner, the BRDY# signal can be asserted every other clock to achieve the two clock timing for each successive access of the burst.

7.1.2 PIPELINED READ PAGE HIT CYCLE

Figure 22 shows a burst read page hit cycle which is pipelined into another burst read cycle. As explained in Section 5.2, the RAS# signal remains asserted for cycles which occur to the same DRAM page. In this case, the page hit signal available from the page comparison logic in the second clock of the bus cycle determines whether to deassert or keep RAS# asserted. If RAS# stays asserted, the CAS# signals can be asserted in the 2nd T2 or third clock of the bus cycle. With a three clock CAS# to data ready delay, the lead-off cycle for the burst read transfer on a page hit therefore requires five CPU clocks.

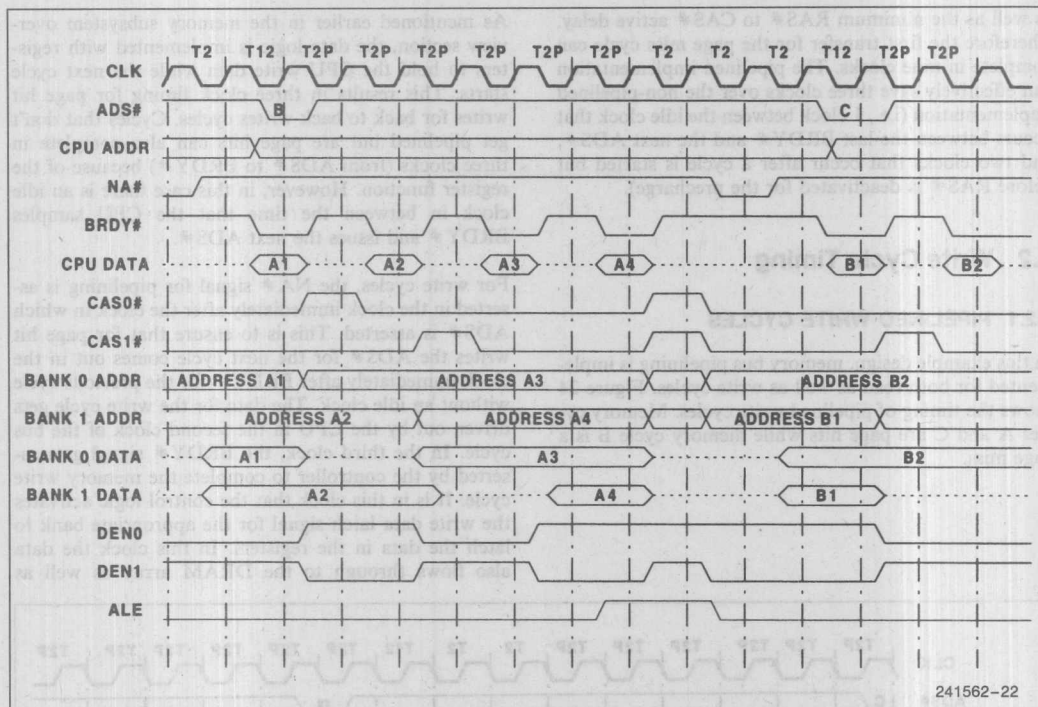


Figure 22. Pipelined Burst Read Page Hit Cycle

With pipelined implementation, the next cycle to the memory subsystem is allowed to start before the previous cycle is completely finished. As shown in the figure, the ADS# signal for memory cycle B is issued by the CPU during memory cycle A. This is in response to the NA# (Next Address) signal being active two clocks earlier. The way the memory subsystem functions during pipelining is that it latches the CPU address and cycle definition information for every cycle. The information gets latched in the second clock of the bus cycle with the ALE signal (generated by address latch enable PLD) going active. The information is kept latched until the end of the cycle. When the CPU issues the next cycle in response to NA#, the DRAM subsystem is able to perform the address decoding and page address comparison for the next cycle while the previous cycle completes. This allows the DRAM transfer corresponding to the pipelined access to be initiated as soon as the last BRDY# for the previous bus cycle is activated. The result is that the first transfer (B1) can complete in four CPU clocks. Each of the subsequent transfers of the burst cycle take 2 clocks. This effectively saves two clocks over the non-pipelined case (i.e., one clock for the idle clock that occurs between the last BRDY# and next ADS#, and one clock to issue the ADS#).

Figure 22 also shows the switching of the data transceivers during burst read cycles. DEN0 is low during data access from DRAM bank 0 while DEN1 is low during data access from DRAM bank 1. The enabling/disabling of the transceivers is handled by the data transceiver enable PLD. Note that the data access A4, for memory cycle A and data access B1, for memory cycle B occur to the same bank. This requires a minimum of four clocks, 1 clock to precharge CAS#, and three more clocks to complete the data transfer.

7.1.3 PIPELINED READ PAGE MISS CYCLE

A burst read page miss cycle is illustrated in Figure 23. The memory cycle is shown pipelined into a previous burst read cycle B. The difference here is the behavior of the RAS# signal. Since cycle C occurs to a different DRAM page than cycle B (determined by the page hit signal), the RAS# signal is deasserted as soon as cycle B completes. This allows the new row address to flow through to the DRAMs. The timing for the page miss cycle adds three clocks for the RAS# precharge time

as well as the minimum RAS# to CAS# active delay. Therefore the first transfer for the page miss cycle can complete in nine clocks. The pipelined implementation can effectively save three clocks over the non-pipelined implementation (i.e., 1 clock between the idle clock that occurs between the last BRDY# and the next ADS#, and two clocks that occur after a cycle is started but before RAS# is deactivated for the precharge).

7.2 Write Cycle Timing

7.2.1 PIPELINED WRITE CYCLES

In this example design, memory bus pipelining is implemented for both read as well as write cycles. Figure 24 shows the timing of pipelined write cycles. Memory cycles A and C are page hits while memory cycle B is a page miss.

As mentioned earlier in the memory subsystem overview section, the data logic is implemented with registers to hold the CPU write data while the next cycle starts. This results in three clock timing for page hit writes for back to back writes cycles. Cycles that don't get pipelined but are page hits can also complete in three clocks (from ADS# to BRDY#) because of the register function. However, in this case there is an idle clock in between the time that the CPU samples BRDY# and issues the next ADS#.

For write cycles, the NA# signal for pipelining is asserted in the clock immediately after the clock in which ADS# is asserted. This is to ensure that for page hit writes the ADS# for the next cycle comes out in the clock immediately after BRDY# for the previous cycle without an idle clock. The data for the write cycle gets driven out by the CPU in the second clock of the bus cycle. In the third clock, the BRDY# signal gets asserted by the controller to complete the memory write cycle. It is in this clock that the control logic activates the write data latch signal for the appropriate bank to latch the data in the registers. In this clock the data also flows through to the DRAM array as well as

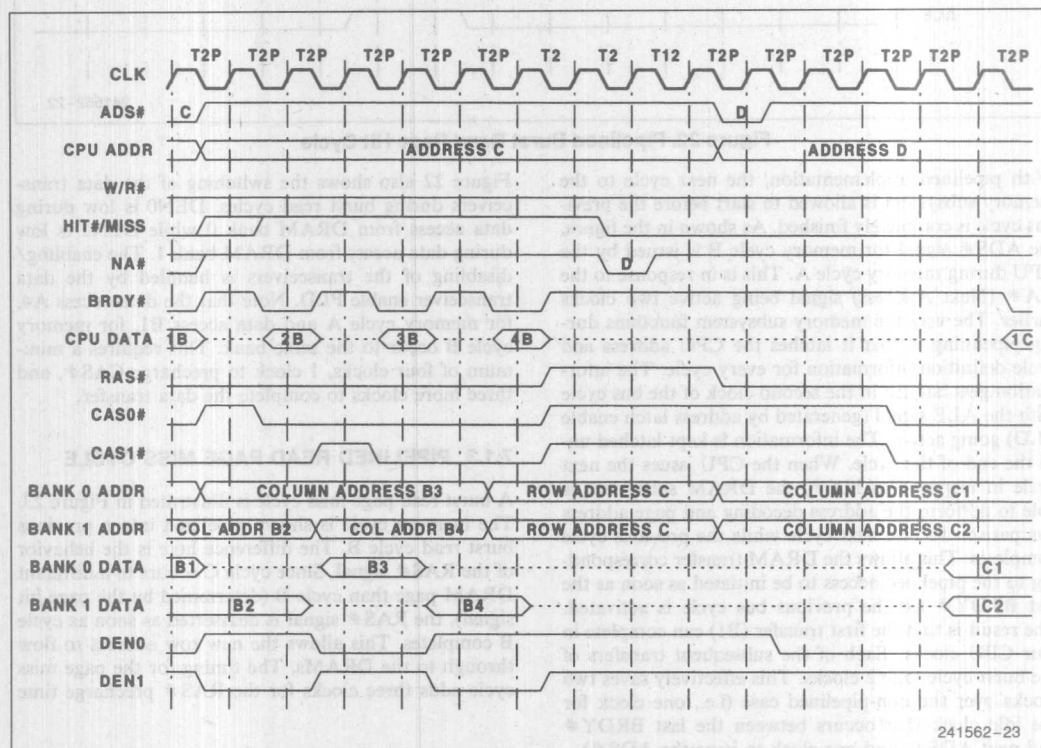


Figure 23. Pipelined Read Page Miss Cycle

CAS# signal is being precharged from the previous write. The actual write to DRAM occurs in the fourth clock. It takes one more clock for a total of five clocks to complete the write transfer to DRAM. However, due to pipelining, cycles are overlapped and therefore to the CPU it appears as a three clock write cycle.

For page miss write cycles, however, the BRDY# activation to the CPU is delayed four more clocks to allow for the RAS# precharge and RAS# to CAS# active delay for write cycles. As shown in Figure 24, cycle B is a page miss; therefore, the page hit signal goes inactive in the second clock. In response to page hit signal inactive, RAS# is deasserted in the third clock and kept inactive for three CPU clocks to meet the RAS# precharge time. The DCP1 signal, to latch the CPU write data, is activated in the third clock and remains active until the data actually gets written to the DRAMs. Since the page miss write cycle is pipelined into a prev-

ious write, the cycle is overlapped with the previous cycle and completes in seven clocks to the CPU.

7.2.2 BURST WRITE CYCLES

Figure 25 shows the timing for a burst write cycle. The Pentium processor supports burst writes only for write-back cycles. The timing for the first transfer of the write cycle is identical to that shown in Figure 24. Since the DRAM array is two-bank interleaved, subsequent transfers can complete in two clocks. A restriction placed by the Pentium processor for burst write cycles is that they cannot be pipelined into other cycles and other cycles are not pipelined into them. The other thing to note for burst writes is that the data transfer always starts with address line 3 being zero (i.e., first transfer always occurs to bank 0).

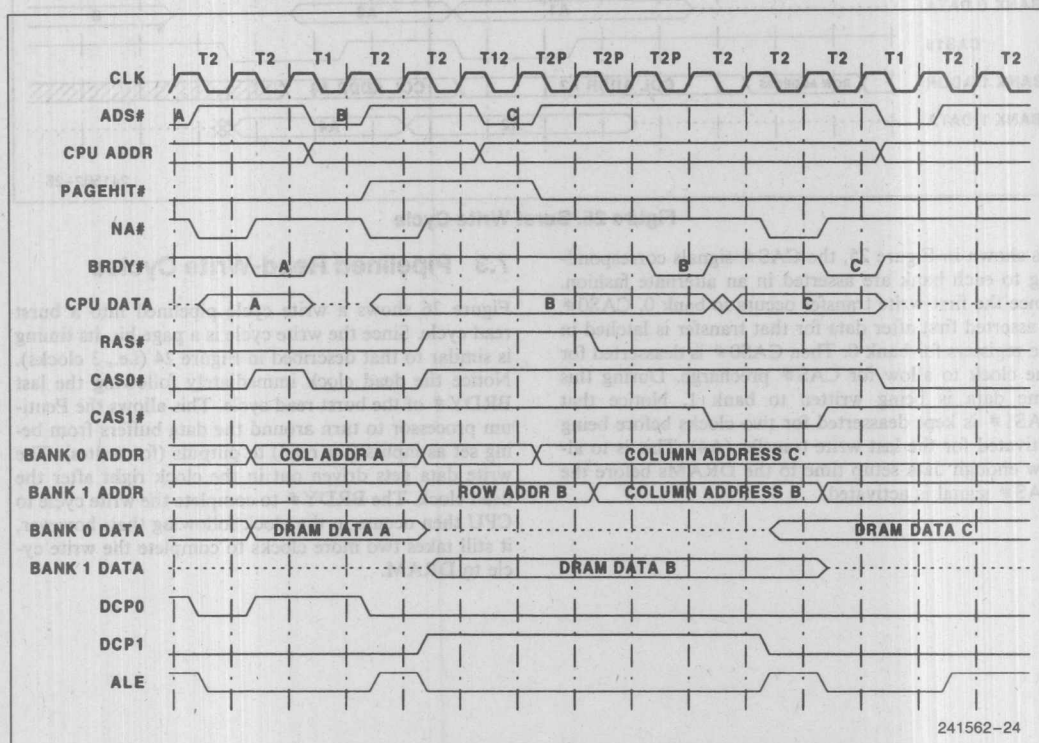


Figure 24. Pipelined Write Cycles

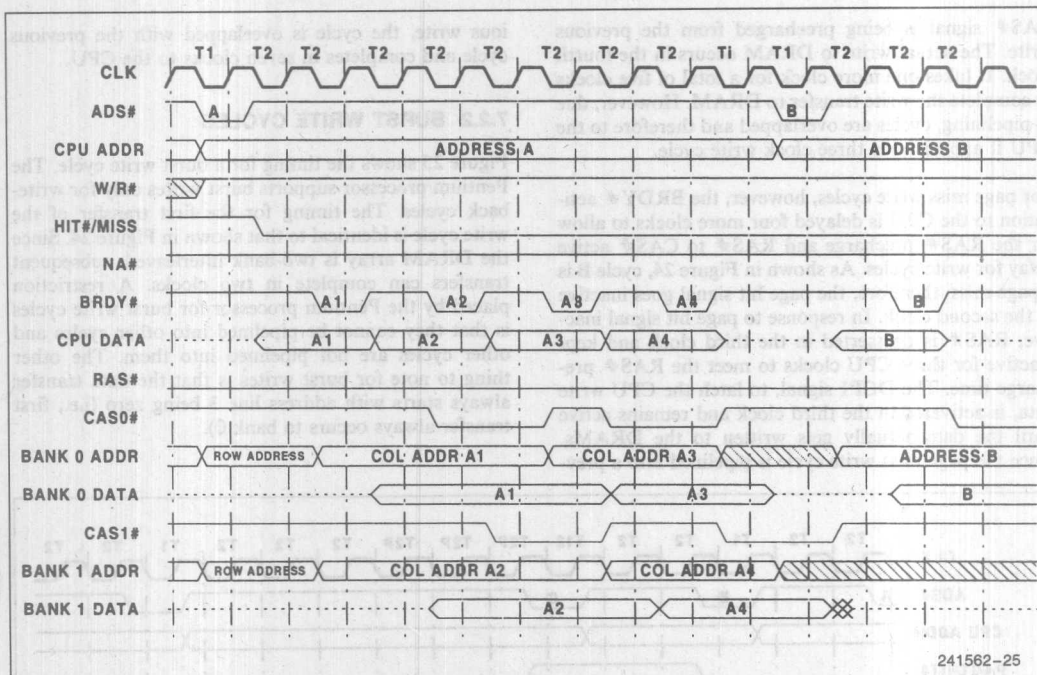


Figure 25. Burst Write Cycle

As shown in Figure 25, the CAS# signals corresponding to each bank are asserted in an alternate fashion. Since the first write transfer occurs to bank 0, CAS0# is asserted first after data for that transfer is latched in the registers for bank 0. Then CAS0# is deasserted for one clock to allow for CAS# precharge. During this time data is being written to bank 1. Notice that CAS1# is kept deasserted for two clocks before being activated for the last write transfer (A4). This is to allow enough data setup time to the DRAMs before the CAS# signal is activated.

7.3 Pipelined Read-Write Cycles

Figure 26 shows a write cycle pipelined into a burst read cycle. Since the write cycle is a page hit, its timing is similar to that described in Figure 24 (i.e., 3 clocks). Notice the dead clock immediately following the last BRDY# of the burst read cycle. This allows the Pentium processor to turn around the data buffers from being set as inputs (for read) to outputs (for writes). The write data gets driven out in the clock right after the dead clock. The BRDY# to complete the write cycle to CPU then occurs in the clock following that; however, it still takes two more clocks to complete the write cycle to DRAM.

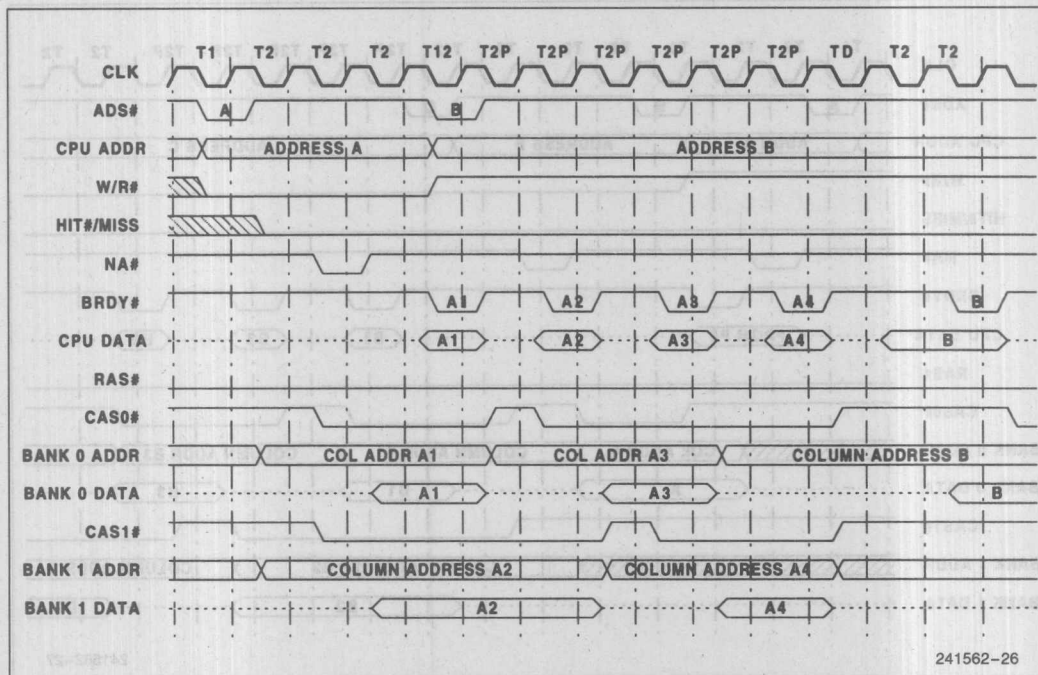


Figure 26. Write Cycle Pipelined into a Burst Read Cycle

7.4 Pipelined Write-Read Cycles

Figure 27 shows a burst read cycle (cycle B) pipelined into a write cycle (cycle A). In this case, two extra clocks are added to the lead-off cycle for the pipelined burst read (i.e., it takes 6 clocks instead of 4 clocks).

The reason is that a write cycle requires two more clocks to complete at the DRAM after the BRDY# is returned to the CPU as explained earlier. If the burst read was a page miss cycle, the number of clocks required for the lead-off cycle would change from 9 clocks to 11 clocks.

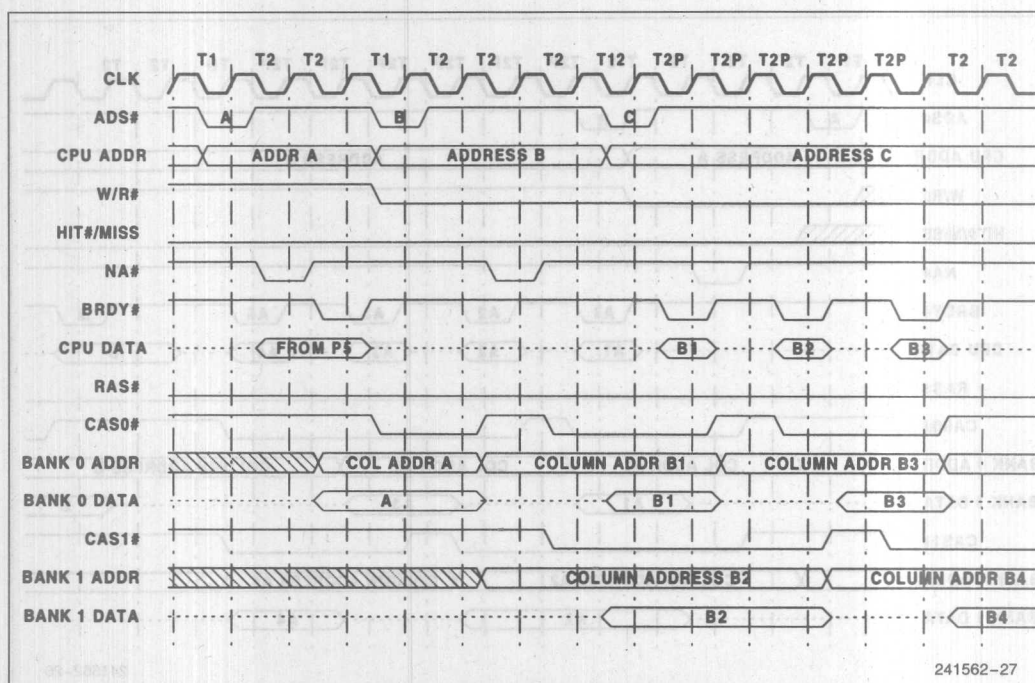


Figure 27. Burst Read Cycle Pipelined into Write Cycle

7.5 Snoop Cycles

Snoop cycles are initiated to the CPU in response to memory bus activity generated by other bus masters in the system such as EISA, ISA, or DMA. In this example the processor is always on HOLD whenever another master is accessing the memory subsystem. The AHOLD signal is not required, which is typically used for running inquire cycles. As soon as EISA/ISA/DMA activity is detected on the processor bus, the EADS# signal is generated to the CPU to initiate the snooping. For EISA/ISA/DMA write cycles to DRAM, the EADS# signal is generated by the EISA Bus Controller. However, for read cycles, the EISA/ISA cycle tracking logic (as part of the memory controller) generates the EADS# signal appropriately. The snoop address also does not require any special handling since all address lines are reflected on the address bus during this time.

Figure 28 shows a CPU writeback due to a snoop hit to a modified cache line during an EISA read cycle. As soon as the EISA cycle start is detected, the EADS# signal is generated to the CPU. At the same time, the

EXRDY# signal is also asserted. The EXRDY# signal drives the output enable of an 'F125 buffer. Since the EISA control signal, EXRDY is open collector, the assertion of EXRDY# pulls the EXRDY signal low. This is done to add wait states to the EISA cycle in case of a snoop hit.

In response to the EADS# signal, the CPU generates the HITM# signal after two clocks to indicate that the snoop has hit a modified line and the CPU needs to issue a writeback cycle. If the HITM# signal is not asserted after two clocks, the EXRDY# signal is deasserted to allow the EISA cycle to continue without wait states.

When the control logic detects the HITM# signal active, the HOLD signal is deasserted to the CPU to allow the CPU to writeback the cache line. The ADS# for the writeback cycle comes out as soon as HLDA is deasserted. The deactivation of HLDA also causes the output buffers on the address and data bus conversion logic (CPU bus to 82350 Host Bus) to turn off to allow the CPU to drive the address and data bus.

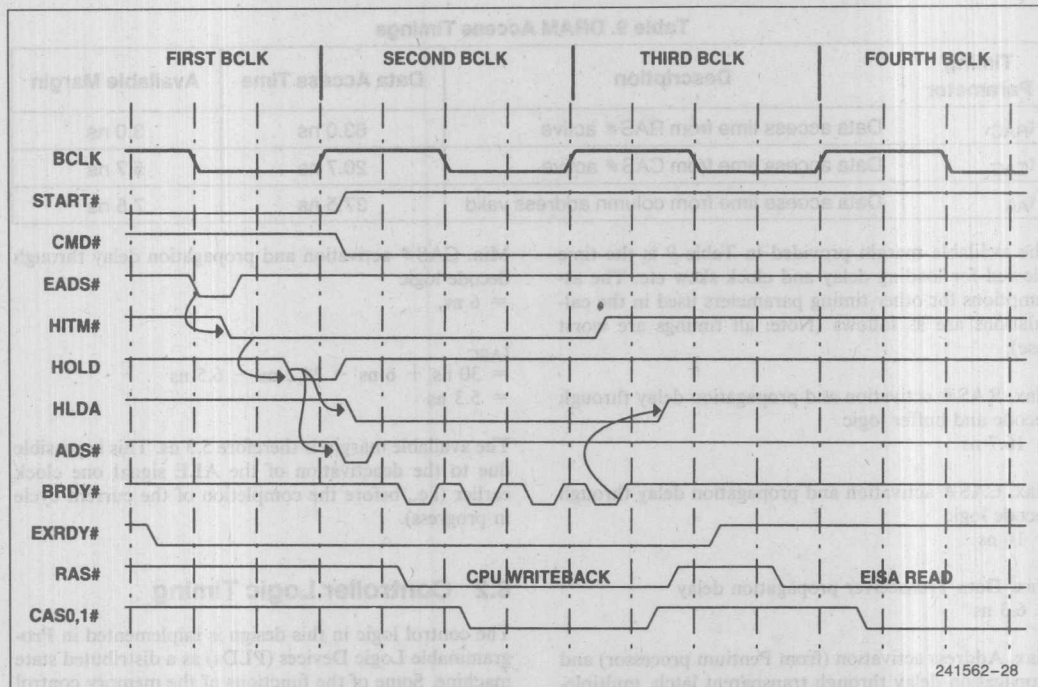


Figure 28. Snoop Cycle—CPU Writeback during EISA Cycle

As soon as the CPU starts the writeback cycle, HOLD is asserted back to the CPU in order to prevent the CPU from issuing another cycle immediately after the writeback completes and to allow the EISA read cycle to complete. Once the CPU has written back the cache line, it asserts the HLDA signal. The EXRDY# signal is also deasserted in order for the EISA read cycle to continue.

The same sequence is followed during an EISA write cycle to memory except that the INV signal is generated to the CPU along with the EADS# signal. This is done in order to invalidate the cache line after it has been written back to memory. During EISA/ISA read cycles, the INV signal is kept inactive to prevent the line from getting invalidated. In this case, only the final state of the cache line is affected (i.e., it is left in the shared state).

Snoop cycles that are run in response to ISA activity on the memory bus follow the same pattern as EISA cycles. The CHRDY# signal in this case is asserted to drive the ISA control signal, CHRDY, low to add wait states to the ISA cycle in case of a snoop hit.

8.0 CRITICAL TIMING CONSIDERATIONS

The control logic for the example memory subsystem is designed to operate synchronously with the CPU clock. At the 66-MHz clock frequency, the timing of some of the signal paths become critical. These critical timing requirements must be met for synchronous operation at 66 MHz. The other timing constraints are dependent on DRAM access timings. These constrain the timing of the memory subsystem to a minimum number of wait states.

8.1 DRAM Access Timing

The DRAM access timing allowed by the memory controller logic is shown in Table 9 (Note: 60-ns DRAMs are used in this design):

Table 9. DRAM Access Timings

Timing Parameter	Description	Data Access Time	Available Margin
t_{RAC}	Data access time from RAS# active	63.0 ns	3.0 ns
t_{CAC}	Data access time from CAS# active	20.7 ns	5.7 ns
t_{AA}	Data access time from column address valid	37.5 ns	7.5 ns

The available margin provided in Table 9 is the time allowed for loading delay and clock skew etc. The assumptions for other timing parameters used in the calculations are as follows (Note: all timings are worst case):

Max. RAS# activation and propagation delay through decode and buffer logic
= 16.7 ns

Max. CAS# activation and propagation delay through decode logic
= 14 ns

Max. Data Transceiver propagation delay
= 6.3 ns

Max. Address activation (from Pentium processor) and propagation delay through transparent latch, multiplexor and buffer logic
= 27.2 ns

Min. Pentium processor Data Setup Time
= 4 ns

Another timing parameter t_{ASC} (Column Address Setup Time to CAS# active) becomes critical during pipelined page hit reads (see Section 7.0, Figure 22). In order to support 4 clock timing for the first access on a pipelined page hit read, the CAS# signal is activated in the second clock of the bus cycle (after the previous cycle completes). The column address for the first access must therefore be available before CAS# is activated in order to initiate the DRAM access. To support this, the memory controller asserts the ALE signal in the last clock of the previous bus cycle to allow enough time for the new address to flow through to the address multiplexors for the next bus cycle. The timing calculation for this parameter is as follows:

Max. ALE deactivation delay (through PLD 13)
= 6.5 ns,

Max. Address propagation delay through transparent latch, multiplexor and buffer logic
= 24.2 ns,

Min. CAS# activation and propagation delay through decode logic
= 6 ns,

t_{ASC}
= 30 ns + 6 ns - 24.2 ns - 6.5 ns
= 5.3 ns

The available margin is therefore 5.3 ns. This is possible due to the deactivation of the ALE signal one clock earlier (i.e., before the completion of the current cycle in progress).

8.2 Controller Logic Timing

The control logic in this design is implemented in Programmable Logic Devices (PLDs) as a distributed state machine. Some of the functions of the memory control logic involve critical signal paths at 66 MHz. These include the PLDs that directly interface with the Pentium processor to generate control signals for the DRAM subsystem or involved in input signal paths to the CPU. The other functions that require faster timing include page hit comparison, RAS# decode and data path switching. The logic for all other functions in the memory subsystem is implemented with 7.5-ns rated programmable logic.

There are four PLDs involved in directly sampling signals generated by the Pentium processor. The signals ADS#, CACHE#, W/R#, and PCHK# must meet the minimum setup time requirement for the PLDs that sample them in the clock in which they are generated (Note: Maximum clock to output valid delay including signal flight time for Pentium processor generated signals plus minimum setup time for PLDs that sample them and change state at the clock edge must be less than the time between clock edges (i.e., 15 ns @ 66 MHz).) Since this particular timing cannot be satisfied by 7.5 ns PLDs, the minimum setup time requirements for PLD 1 (CPU Cycle Tracking Logic), PLD 13 (ALE Generation), PLD 15 (NA# generation) and PLD 22 (Synchronize Pentium processor signals for 33-MHz EBC) can be met by using 5-ns programmable logic devices.

include PLD 9 (BRDY# generate for read/write cycles), PLD 23 (BRDY# combine) and PLD21 (Generates EADS#, FLUSH#, HOLD to Pentium processor) which require faster propagation delay (t_{PD}) or faster clock to output delay (t_{CO}) in order to satisfy the minimum input setup time requirements for the Pentium processor. The margin available on the BRDY# input to the CPU using 5-ns programmable devices is as follows:

Margin available on BRDY#

- = (1 CLK)
- (Time to generate and combine BRDY#)
- (Minimum Pentium processor BRDY# Setup Time)
- = 15 ns (4 ns + 5 ns) 4 ns
- = 2 ns

For page address comparison, PLD 9 (Page hit detect and row decode) generates the page hit signal which must be available for sampling by the PLDs in the second level logic by the end of the second clock of the bus cycle. The margin available for signal flight time (for CPU address lines) and clock skew on the page hit signal is as follows (see Section 5.2 for description of page address comparison logic):

Margin on Page Hit signal

- = 2 CLKs
- (Maximum Pentium processor Address Valid Delay)
- ('521B Comparator Propagation Delay)
- (Maximum Prop. Delay through PLD 9)
- (Minimum PLD setup time)
- = 30 ns - 8 ns - 5.5 ns - 7.5 ns - 7 ns
- = 2 ns

The above calculation assumes using a 7.5 ns PLD for PLD 9. By implementing the page hit detect logic in a 5 ns programmable device, the available margin may be increased to 4.5 ns.

During burst read cycles, the data transfer is alternated between the two banks. The margin available on the data path for each bank is as follows:

Margin on Data Path Enable

- = 2 CLKs
- (Max. delay to activate DEN0/1 from PLD 10)
- (Max. Data Transceiver, '646A output enable time)
- (Min. Pentium processor Data Setup Time)
- = 30 ns - 13.5 ns - 9.8 ns - 4 ns
- = 2.7 ns

read cycles is 2.7 ns and therefore requires the data transceiver enable function to be implemented in 5 ns programmable logic.

9.0 SUMMARY

This report presented performance trade-offs in memory subsystem design for a Pentium processor-based system. It also discussed the design of an example memory subsystem without a second level cache for the Pentium processor.

The following can be concluded from the material presented in the memory performance section:

- The Pentium processor bus interface is different than that for the Intel486 CPU. It is not dominated by write cycles (as was the case with the Intel486 CPU), due to a write-back cache architecture for its data cache. The bus cycle mix and other simulation results showed that optimizations in read cycle timing was more important than write cycle timing for the Pentium processor.
- Among the read cycle timing parameters, the read burst timing showed more sensitivity to wait states than read lead off cycle timing. Approximately 3%–5% loss in overall performance can result from every wait state added to read burst timing.
- Optimization in write burst timing is not very critical for Pentium processor desktop systems as it constitutes only a small percentage of overall bus activity.
- Implementing DRAM subsystem designs to include two bank interleaving and DRAM paging will add 5%–10% to overall performance. This is important especially for systems that do not incorporate a second level cache.
- Poor choices of bus speed, bus width and bus pipelining can degrade overall performance by 25%–35% in Pentium processor-based L2 cacheless systems.

The design example also demonstrated some key functions of Pentium processor memory subsystem design. These included writeback support for Pentium processor in the DRAM subsystem; cycle pipelining for read and write cycles; two bank interleaving; paging, etc. The example also detailed the interface logic necessary to interface to a half speed, 32-bit bus. The simulated performance of this memory subsystem was found to range between 80%–94% for different applications relative to an ideal zero wait state system.

APPENDIX A PERFORMANCE SIMULATION METHODOLOGY

The simulation data presented in this report was extracted from an Intel internal performance simulator. The simulator has the capability to simulate a Pentium processor, an L2 cache and a memory system.

It also features the ability to vary memory subsystem parameters such as memory cycle timing, paging, pipelining, bus size, bus frequency and the ability to vary cache parameters for analyzing different hardware configurations. For the simulation results presented here, traces from five different applications running under DOS, UNIX and Windows were used. All the traces are random samples of instructions which were captured while the respective applications were running. The DOS and UNIX traces are hardware generated. On the other hand, the Windows traces are software generated.

In the hardware tracing scheme, the application is allowed to run in real time, while connected to a special bus monitoring hardware which stores all the bus activity for some period of time. After the bus activity has been collected, it is then processed to infer the state of the CPU during the trace. The hardware tracing scheme does not have access to the internal state of the CPU while the application is running; however, it does execute in real-time and therefore includes operating system calls, I/O activity, timer interrupts, etc.

The software tracing scheme, such as that used for Windows applications, were collected using the Micro-

soft Windows Kernel Debugger. This software debugger/tracer single steps through every instruction in an application and dumps some state information after executing each instruction. The application being traced runs on a PC. This machine dumps the trace information to a serial port connected to another PC which captures and stores the trace. The advantage of the software tracing scheme is the ability to access the CPU state which cannot be inferred just by monitoring external bus activity.

For information on acquiring a simulator and Pentium processor models to run simulations similar to those presented in this report, the following companies may be contacted.

For the simulator, contact

SES (Scientific and Engineering Software, Inc.)
4301 Westbank Drive
Austin, Texas 78746
Tel: (512) 328-5544
Fax: (512) 327-6646

For Pentium processor models, contact

C.A.E. Plus, Inc.
9300 Jollyville Rd., #106
Austin, Texas 78759
Tel: (512) 338-0165
Fax: (512) 338-0192

APPENDIX B PARTS LIST

The following is a parts list for the example memory subsystem. Note only the major components used in the DRAM subsystem and bus conversion logic are shown.

Component	Quantity
Clock Driver 66-MHz Clock Driver—Triquint GA1086	1
DRAM Address Latches 74FCT573 (Octal Transparent Latch)	4
DRAM Address Multiplexors 74FCT257 (Quad 2 to 1 Multiplexors)	4
DRAM Data Transceivers 74FCT646A (Octal Transceiver/Register)	18
Driver Buffers 74F244 (Octal Buffers)	7
DRAM Modules 256K x 36 or 512K x 36 (72-pin SIMM)	4-8
Address Decode SRAM MT5C6408-10 (8K x 8 SRAM)	1
CPU to EBC Host Bus Address Buffers 74FCT645 (Octal Transceivers)	4
Byte Enable and Data Parity Conversion 85C060 (16 Macrocell PLD)	2
CPU to EBC Host Bus (32-bit) Data Bus Buffers 82352 (EISA Bus Buffers)	2
Page Address Comparison Logic 74FCT521B (8-bit Identity Comparator)	1
74F273 (Octal D-Flip Flop)	2
DRAM Control/Bus Conversion Logic (including EISA/ISA memory cycle tracking and address decode) 85C220/85C224/85C22V10 PLDs	24

3

APPENDIX C

PLD CODES AND SCHEMATICS

Attached are the PLD codes and schematics that cover the example Pentium Processor Memory Subsystem and Bus Conversion Logic. Note: the schematics cover only the DRAM subsystem and the bus conversion logic to interface to the 82350 EISA Chip Set. Details on EISA bus and XBUS interface with the 82350 Chip Set

can be found in the *Intel486™ Microprocessor-Based 82350 EISA Design Guide*, Order No. 296504-001.

The following is a reference list of the PLDs used in the memory subsystem and bus conversion logic.

PLD No.	Function	Device Type
1	CPU Cycle Tracking Logic	N85C22V10
2	RAS Generator	N85C22V10
3	CAS Generator for Bank 0	N85C22V10
4	CAS Generator for Bank 1	N85C22V10
5	RAS Decode/RAS Precharge Count	N85C220
6	CAS Decode Bank 0	N85C224
7	CAS Decode Bank 1	N85C224
8	Burst Ready Generation for Read/Write	N85C22V10
9	Page Hit Comparison and Row Decode	N85C220
10	Data Transceiver Enable Bank 0/1	N85C224
11	Activate Write Data Register Bank 0/1	N85C220
12	Bank Select/Burst Count/Row Addr Latch Enable	N85C220
13	Address Latch Enable for Cycle Pipelining	N85C22V10
14	Burst Address Generation	N85C224
15	Next Address Signal Generator	N85C224
16	Pentium Processor/EBC Byte Enable Generation/Buffer Logic	N85C060
17	ISA Cycle Tracking Logic	N85C220
18	EISA Cycle Tracking Logic	N85C22V10
19	Pentium Processor/EBC Data Parity Generation/Buffer Logic	N85C060
20	Bank Select during EISA Burst Cycles	N85C224
21	Generates Input Signals for Pentium Processor	N85C220
22	Synchronize Pentium Processor Signals for 33-MHz EBC	N85C220
23	Combine BRDY # and Generate CPURDY # for EBC	N85C220
24	Write Enable Generation PLD	N85C224

module PLD1

title 'DRAM CONTROLLER - PLD 1
INTEL CORPORATION, August 1992'

- CPU Cycle Tracking Logic - This PLD generates the appropriate signals for tracking CPU cycles to DRAM. It also generates the KEN# signal for the CPU.

PLD1 device 'P22V10C';

x, c, z = .X., .C., .Z.;

* Inputs

CLK	pin	2;	"CPU input clock"
ADSn	pin	3;	"CPU ADS# Output"
DRAMCSn	pin	4;	"DRAM Chip Select"
BCNT	pin	5;	"Burst Count Active with 4th BRDY for Burst cycles"
NAn	pin	6;	"CPU Next Address Input"
WRn	pin	7;	"CPU Write/Read Indication"
PCACHEn	pin	9;	"CPU CACHE# output"
HITn	pin	10;	"DRAM Page Hit Indicator"
MRBRDYn	pin	11;	"Burst Ready for Read Cycles"
MWBRDYn	pin	12;	"Burst Ready for Write Cycles"
RESET	pin	13;	"CPU Reset Signal"
HLDA	pin	17;	"CPU Hold Acknowledge Signal"
KENn	pin	27;	"Cache Enable Output from Decode Logic"

* Outputs

CIPn	pin	21;	"CPU DRAM Cycle in Progress"
PIPECYCn	pin	25;	"Active during Pipelined Cycles"
CYC4X	pin	23	istype 'buffer'; "CPU Four Transfer Cycle"
CT	pin	26	istype 'buffer'; "Cycle Track Output for Pipelining"
P5KENn	pin	18	istype 'buffer'; "KEN# input to CPU"
Var1	pin	20	istype 'buffer'; "State Variable"
Var2	pin	24	istype 'buffer'; "State Variable"
Var3	pin	19	istype 'buffer'; "State Variable"

* State Register Assignments

sreg	=	[Var1, Var2, Var3];
S0	=	[0, 0, 0];
S1	=	[0, 0, 1];
S2	=	[0, 1, 1];
S3	=	[1, 1, 1];
S4	=	[1, 0, 1];

state_diagram sreg

state S0 :

if RESET then S0 with CIPn := 1; PIPECYCn := 1; endwith else
if IADSn & IHLDA then S1 with CIPn := 0; PIPECYCn := 1; endwith
else S0 with CIPn := 1; PIPECYCn := 1; endwith;

state S1 :

241562-29

```

if RESET # DRAMCSn # (ICT.FB & BCNT) then S0 with CIPn := 1;
    PIPECYCn := 1; endwith else
if (PCACHEn & !HITn & !WRn) # (PCACHEn & !HITn & WRn) then S2
    with CIPn := 0; PIPECYCn := 1; endwith else
if (IPACHEn & !INAn) # (WRn & IPACHEn) # (WRn & PCACHEn & HITn) then S3
    with CIPn := 0; PIPECYCn := 0; endwith
else S1 with CIPn := 0; PIPECYCn := 1; endwith;

state S2 :
if RESET # !MWBRDYN # (!MRBRDYN & ADSn) then S0 with CIPn := 1;
    PIPECYCn := 1; endwith else
if !MRBRDYN & !ADSn then S1 with CIPn := 0; PIPECYCn := 1; endwith else
if !INAn & MRBRDYN & !WRn then S4 with CIPn := 0; PIPECYCn := 0; endwith
else S2 with CIPn := 0; PIPECYCn := 1; endwith;

state S3 :
if RESET # (ICT.FB & BCNT) # (ICT.FB & !CYC4X.FB & !MWBRDYN) then S0
    with CIPn := 1; PIPECYCn := 1; endwith else
if (CT.FB & BCNT) # (CT.FB & !CYC4X.FB & !MWBRDYN) then S1
    with CIPn := 0; PIPECYCn := 1; endwith
else S3 with CIPn := 0; PIPECYCn := 0; endwith;

state S4 :
if RESET # (!MRBRDYN & ICT.FB) then S0 with CIPn := 1;
    PIPECYCn := 1; endwith else
if (!MRBRDYN & CT.FB) then S1 with CIPn := 0; PIPECYCn := 1; endwith
else S4 with CIPn := 0; PIPECYCn := 0; endwith;

state_diagram [CT]

state [0] : if RESET then [0] else
    if (!HLDA & !ADSn & !PIPECYCn) then [1] else [0];
state [1] : if RESET then [0] else
    if (!CIPn & PIPECYCn)
        # (!MWBRDYN & !CYC4X.FB & !HITn) then [0] else [1];

state_diagram [P5KENn]

state [1] : if RESET then [1] else
    if (!HLDA & !CIPn & !DRAMCSn & !KENn & !WRn) then [0] else [1];
state [0] : if RESET then [1] else
    if CIPn # (KENn & !DRAMCSn) then [1] else [0];

equations

[Var1, CIPn, PIPECYCn, Var2, Var3, CT, P5KENn, CYC4X].clk = CLK;
[Var1, Var2, Var3, CT, P5KENn, CYC4X].AR = RESET;

CIPn.OE      =      !HLDA;
CYC4X :=      (!PCACHEn) # (CYC4X.FB & !PIPECYCn);

end PLD1;

```

241562-30

module PLD2

title 'DRAM CONTROLLER - PLD 2
INTEL CORPORATION, August 1992'

" RAS Generator PLD - This PLD generates the DRAM Row Address Strobe
Signal (RAS#)

PLD2 device 'P22V10C';

x, c, z = .X., .C., .Z.;

Inputs

CLK	pin	2;	"CPU input clock"
CIPn	pin	3;	"DRAM Cycle in Progress"
DRAMCSn	pin	4;	"DRAM chip select"
CT	pin	5;	"CPU Cycle Track"
HITn	pin	6;	"DRAM Page Hit Signal"
BCNT	pin	9;	"Burst Count active with 4th BRDY for Burst Cycles"
MRBRDYN	pin	10;	"Burst Ready Input for Read Cycles"
RESET	pin	11;	"CPU Reset Signal"
HLDA	pin	12;	"CPU Hold Acknowledge Signal"
WRn	pin	13;	"Write/Read signal"
PCHG	pin	16;	"RAS Precharge Count"
WOn	pin	25;	"Write Cycle Indication for Bank 0"
W1n	pin	19;	"Write Cycle Indication for Bank 1"
MION	pin	7;	"Memory/I/O signal"

Outputs

RASn	pin	20	istype 'buffer'; "DRAM Row Address Strobe Signal"
Var1	pin	26	istype 'buffer'; "State Variable"
Var2	pin	18	istype 'buffer'; "State Variable"
DMEMR	pin	24;	"DRAM read cycle indicator"
DMEMW	pin	23;	"DRAM write cycle indicator"
WIPn	pin	21;	"Write Cycle in Progress"

State Register Assignments

```
sreg = [Var1, Var2];
S0 = [0, 0];
S1 = [0, 1];
S2 = [1, 1];
S3 = [1, 0];
```

state_diagram sreg

state S0 :

```
if RESET then S0 with RASn := 1; endwith else
if !CIPn & !PCHG & !DRAMCSn then S1 with RASn := 0; endwith
else S0 with RASn := 1; endwith;
```

state S1 :

```
if RESET # (!CIPn & !DRAMCSn & HITn & !CT & !HLDA) # (HLDA & CIPn)
# (!HLDA & CT & !WIPn & HITn) then S0 with RASn := 1; endwith else
if !HLDA & CT & HITn & !CIPn & !DRAMCSn & !WRn & WIPn then S2
with RASn := 0; endwith
```

241562-31


```
else S1 with RASn := 0; endwith;
```

```
state S2 :
```

```
if RESET # (BCNT & IMRBRDYn) then S0 with RASn := 1; endwith
else S2 with RASn := 0; endwith;
```

```
equations
```

```
[Var1, RASn, Var2].clk = CLK;
```

```
[Var1, RASn, Var2].AR = RESET;
```

```
DMEMR = (!CIPn & MIOn & !WRn & !HLDA & !DRAMCSn)
          # (!CIPn & HLDA & !WRn);
```

```
DMEMW = (!CIPn & MIOn & WRn & !HLDA & !DRAMCSn)
          # (!CIPn & HLDA & WRn);
```

```
!WIPn = !W0n # !W1n;
```

```
end PLD2;
```

241562-32

module PLD3

title 'DRAM CONTROLLER - PLD 3
INTEL CORPORATION, August 1992'

- * CAS Generator PLD for Bank 0 - This PLD generates the master DRAM Column
- * Address Strobe Signal for Bank 0. It also generates the Write Cycle Active indication for
- * Bank 0.

PLD3 device 'P22V10C';

x, c, z = .X., .C., .Z.;

Inputs

CLK	pin	2;	"CPU input clock"
CIPn	pin	3;	"DRAM Cycle in Progress"
CYC4X	pin	4;	"CPU four transfer cycle"
HITn	pin	5;	"DRAM Page Hit Signal for CPU cycles"
BCNT	pin	6;	"Burst Count active with 4th BRDY for Read Cycles"
MRBRDYn	pin	7;	"Burst Ready Input for Read Cycles"
MWBRDYn	pin	9;	"Burst Ready Input for Write Cycles"
RESET	pin	10;	"CPU Reset Signal"
HLDA	pin	11;	"CPU Hold Acknowledge Signal"
WRn	pin	12;	"Read/Write Cycle Indicator"
DRAMCSn	pin	13;	"DRAM Chip Select"
RASn	pin	16;	"Row address strobe signal"
REFRESHn	pin	17;	"Refresh sequence indicator"
BANKSEL	pin	18;	"Banksel indicates Bank 0 or Bank 1"
CT	pin	27;	"CPU Cycle Track for pipelined cycles"
ALE	pin	26;	"Address Latch Enable Signal"

Outputs

!CAS0n	pin	23	istype 'invert'; "CAS Signal for Bank 0"
!W0n	pin	20	istype 'invert'; "Active for Write Cycles to Bank 0"
Var1	pin	24	istype 'buffer'; "State Variable"
Var2	pin	25	istype 'buffer'; "State Variable"
Var3	pin	21	istype 'buffer'; "State Variable"

State Register Assignments

sreg	=	[CAS0n, W0n, Var1, Var2, Var3];
S0	=	[0, 0, 0, 0, 0]; "CAS0n = 1; W0n = 1; 0
S1	=	[0, 0, 0, 0, 1]; "CAS0n = 1; W0n = 1; 1
S2	=	[0, 0, 0, 1, 0]; "CAS0n = 1; W0n = 1; 2
S3	=	[1, 0, 0, 0, 0]; "CAS0n = 0; W0n = 1; 0
S4	=	[1, 0, 0, 0, 1]; "CAS0n = 0; W0n = 1; 1
S5	=	[0, 0, 0, 1, 1]; "CAS0n = 1; W0n = 1; 3
S6	=	[1, 0, 0, 1, 0]; "CAS0n = 0; W0n = 1; 2
S7	=	[0, 1, 0, 0, 0]; "CAS0n = 1; W0n = 0; 0
S8	=	[1, 1, 0, 0, 0]; "CAS0n = 0; W0n = 0; 0
S9	=	[1, 1, 0, 0, 1]; "CAS0n = 0; W0n = 0; 1
S10	=	[0, 1, 0, 0, 1]; "CAS0n = 1; W0n = 0; 1
S11	=	[0, 1, 0, 1, 0]; "CAS0n = 1; W0n = 0; 2
S12	=	[0, 0, 1, 0, 0]; "CAS0n = 1; W0n = 1; 4

state_diagram sreg

241562-33

```

state S0 :
    if REFRESHn & !CIPn & !DRAMCSn & !WRn & !RASn & !ALE &
        (!BANKSEL & CYC4X) then S3
    else if REFRESHn & !CIPn & !DRAMCSn & !WRn & RASn then S1
    else if REFRESHn & !CIPn & !DRAMCSn & WRn then S7
    else S0;

state S1 :
    if (!HLDA & HITn) # (BANKSEL & !CYC4X) # WRn #
        (BANKSEL & HLDA) then S0
    else if (!BANKSEL & !CT & !RASn) # (!HLDA & !RASn & !CT & CYC4X) then S2
    else if (!BANKSEL & CT & !RASn) # (!HLDA & !RASn & CT & CYC4X) then S3
    else S1;

state S2 :
    goto S12;

state S3 :
    if (!CYC4X & !MRBRDYn & !HLDA) then S0
    else if !MRBRDYn & !BANKSEL & !HLDA then S5
    else if (!MRBRDYn & !BANKSEL) # HLDA then S4
    else S3;

state S4 :
    if !MRBRDYn & !HLDA then S5
    else if HLDA then S6
    else S4;

state S5 :
    goto S6;

state S6 :
    if (BCNT & !CT) # HLDA then S0
    else if (BCNT & CT & !HLDA) then S1
    else S6;

state S7 :
    if (BANKSEL & !CYC4X) then S0
    else if !HLDA & !RASn & !CYC4X & !MRBRDYn & !BANKSEL then S9
    else if (!RASn & CYC4X & !MRBRDYn) # (!RASn & HLDA) then S10
    else S7;

state S8 :
    if HLDA # CIPn # DRAMCSn # !WRn # BANKSEL then S0
    else S11;

state S9 :
    goto S8;

state S10 :
    goto S9;

state S11 :
    if RASn then S7
    else if !MRBRDYn then S9
    else if CIPn then S0
    else S11;

```

```
state S12 :
goto S3;
```

equations

```
[Var1, Var2, Var3, CAS0n, W0n].clk = CLK;
[Var1, Var2, Var3, CAS0n, W0n].AR = RESET;
```

end PLD3;

241562-35

3

module PLD4

title 'DRAM CONTROLLER - PLD 4
INTEL CORPORATION, August 1992'

- " CAS Generator PLD for Bank 1 - This PLD generates the master DRAM Column
- " Address Strobe Signal (CAS) for Bank 1. It also generates the Write Cycle Active
- " indication for Bank 1.

PLD4 device 'P22V10C';

x, c, z = .X., .C., .Z.;

Inputs

CLK	pin	2;	"CPU input clock"
CIPn	pin	3;	"DRAM Cycle in Progress"
CYC4X	pin	4;	"CPU four transfer cycle"
HITn	pin	5;	"DRAM Page Hit Signal for CPU cycles"
BCNT	pin	6;	"Burst Count active with 4th BRDY for Burst Cycles"
MRBRDYn	pin	7;	"Burst Ready Input for Read Cycles"
MWBRDYn	pin	9;	"Burst Ready Input for Write Cycles"
RESET	pin	10;	"CPU Reset Signal"
HLDA	pin	11;	"CPU Hold Acknowledge Signal"
WRn	pin	12;	"Read/Write Cycle Indicator"
DRAMCSn	pin	13;	"DRAM Chip Select"
RASn	pin	16;	"Row address strobe signal"
REFRESHn	pin	17;	"Refresh sequence indicator"
BANKSEL	pin	18;	"Banksel indicates Bank 0 or Bank 1 access"
CT	pin	27;	"CPU Cycle Track for Pipelined cycles"
ALE	pin	26;	"Address Latch Enable Signal"

Outputs

!CAS1n	pin	23	istype 'invert'; "CAS Signal for Bank 1"
!W1n	pin	25	istype 'invert'; "Active for Write Cycle to Bank 1"
Var1	pin	24	istype 'buffer'; "State Variable"
Var2	pin	20	istype 'buffer'; "State Variable"
Var3	pin	21	istype 'buffer'; "State Variable"

State Register Assignments

```

sreg = [CAS1n, W1n, Var1, Var2, Var3];
S0 = [0, 0, 0, 0, 0]; "CAS1n = 1; W1n = 1; 0
S1 = [0, 0, 0, 0, 1]; "CAS1n = 1; W1n = 1; 1
S2 = [0, 0, 0, 1, 0]; "CAS1n = 1; W1n = 1; 2
S3 = [1, 0, 0, 0, 0]; "CAS1n = 0; W1n = 1; 0
S4 = [1, 0, 0, 0, 1]; "CAS1n = 0; W1n = 1; 1
S5 = [0, 0, 0, 1, 1]; "CAS1n = 1; W1n = 1; 3
S6 = [1, 0, 0, 1, 0]; "CAS1n = 0; W1n = 1; 2
S7 = [0, 1, 0, 0, 0]; "CAS1n = 1; W1n = 0; 0
S8 = [1, 1, 0, 0, 0]; "CAS1n = 0; W1n = 0; 0
S9 = [1, 1, 0, 0, 1]; "CAS1n = 0; W1n = 0; 1
S10 = [0, 1, 0, 0, 1]; "CAS1n = 1; W1n = 0; 1
S11 = [0, 1, 0, 1, 0]; "CAS1n = 1; W1n = 0; 2
S12 = [0, 0, 1, 0, 0]; "CAS1n = 1; W1n = 1; 4

```

state_diagram sreg

241562-36

```

state S0 :
    if REFRESHn & ICIPn & IDRAMCSn & !WRn & !RASn & !ALE &
      (BANKSEL # CYC4X) then S3 else
    if REFRESHn & ICIPn & WRn & IDRAMCSn then S7 else
    if REFRESHn & ICIPn & IDRAMCSn & !WRn & RASn then S1
    else S0;

state S1 :
    if (!HLDA & HITn) # (!BANKSEL & !CYC4X) # WRn #
      (BANKSEL & HLDA) then S0 else
    if (BANKSEL & !RASn & !CT) # (!HLDA & !RASn & CYC4X & !CT) then S2 else
    if (BANKSEL & !RASn & CT) # (!HLDA & !RASn & CYC4X & CT) then S3
    else S1;

state S2 :
    goto S12;

state S3 :
    if (!CYC4X & !MRBRDYn & !HLDA) then S0 else
    if !MRBRDYn & BANKSEL & !HLDA then S5 else
    if !MRBRDYn & !BANKSEL # HLDA then S4
    else S3;

state S4 :
    if !MRBRDYn & !HLDA then S5 else
    if HLDA then S6
    else S4;

state S5 :
    goto S6;

state S6 :
    if (BCNT & !CT) # HLDA then S0 else
    if (BCNT & CT) then S1
    else S6;

state S7 :
    if (!BANKSEL & !CYC4X) then S0 else
    if !HLDA & !RASn & !CYC4X & !MWBRDYn & BANKSEL then S9 else
    if (!RASn & CYC4X & !MWBRDYn) # (!RASn & HLDA) then S10
    else S7;

state S8 :
    if HLDA # CIPn # !WRn # DRAMCSn
      # (!BANKSEL & !CYC4X) then S0
    else S11;

state S9 :
    goto S8;

state S10 :
    if !MWBRDYn # HLDA then S9
    else S10;

state S11 :
    if RASn then S7
    else if !MWBRDYn then S9
  
```

```

else S11;
state S12 :
goto S3;

```

equations

```

[Var1, Var2, Var3, CAS1n, W1n].clk = CLK;
[Var1, Var2, Var3, CAS1n, W1n].AR = RESET;

```

end PLD4;

241562-38

module PLD5

title 'DRAM CONTROLLER - PLD 5
INTEL CORPORATION, August 1992'

- RAS Decode and RAS Precharge Count PLD - This PLD decodes RAS signals for the DRAM array. It also generates the RAS precharge count signal for miss cycles.

PLD5 device 'E0320';

x, c, z = .X., .C., .Z.;

▪ Inputs

CLK	pin	1;	"CPU input clock"
RESET	pin	4;	"CPU Reset Signal"
ROWAn	pin	5;	"Row A"
ROWBn	pin	6;	"Row B"
HIMEMn	pin	7;	"Upper Half of 512K indicator"
RASn	pin	8;	"Row Address Strobe Signal"

▪ Outputs

PCHG	pin	12;	"RAS Precharge Count"
RASAO _n	pin	13;	"Row Address Strobe for Lower Half of 512K in Row A"
RASA1 _n	pin	14;	"Row Address Strobe for Upper Half of 512K in Row A"
RASBO _n	pin	15;	"Row Address Strobe for Lower Half of 512K in Row B"
RASB1 _n	pin	16;	"Row Address Strobe for Upper Half of 512K in Row B"
Var1	pin	17;	"State Variable"
Var2	pin	18;	"State Variable"

▪ State Register Assignments

sreg	=	[Var1, Var2];
S0	=	[0, 0];
S1	=	[0, 1];
S2	=	[1, 1];
S3	=	[1, 0];

state_diagram sreg

```

state S0 :
    if RESET then S0 with PCHG := 0; endwith else
    if !RASn then S1 with PCHG := 1; endwith
    else S0 with PCHG := 0; endwith;

state S1 :
    if RESET then S0 with PCHG := 0; endwith else
    if RASn then S2 with PCHG := 1; endwith
    else S1 with PCHG := 1; endwith;

state S2 :
    goto S0 with PCHG := 0; endwith;

state S3 :
    goto S0 with PCHG := 0; endwith;
    
```

equations

[Var1, PCHG, Var2].clk = CLK;

IRASA0n = IRASn & HIMEMn & !ROWAn;

IRASA1n = IRASn & HIMEMn & !ROWAn;

IRASB0n = IRASn & HIMEMn & !ROWBn;

IRASB1n = IRASn & HIMEMn & !ROWBn;

end PLD5;

241562-40

module PLD6

title 'DRAM CONTROLLER - PLD 6
INTEL CORPORATION, August 1992'

" CAS Decode PLD for Bank 0"

PLD6 device 'E224C';

x, c, z = .X., .C., .Z.;

" Inputs

CAS0n	pin	27;	"Column Address Strobe for Bank 0"
DMEMR	pin	2;	"DRAM Read Cycle"
DMEMW	pin	3;	"DRAM Write Cycle"
LBE0n	pin	4;	"Latched Byte Enables BE0# - BE7#"
LBE1n	pin	5;	
LBE2n	pin	6;	
LBE3n	pin	7;	
LBE4n	pin	9;	
LBE5n	pin	10;	
LBE6n	pin	11;	
LBE7n	pin	12;	

" Outputs

CAS00n	pin	18;	"CAS Decode outputs for Bank 0"
CAS01n	pin	19;	"
CAS02n	pin	20;	"
CAS03n	pin	21;	"
CAS04n	pin	23;	"
CAS05n	pin	24;	"
CAS06n	pin	25;	"
CAS07n	pin	26;	"

equations

!CAS00n	=	(!CAS0n & DMEMR) # (!CAS0n & DMEMW & !LBE0n);
!CAS01n	=	(!CAS0n & DMEMR) # (!CAS0n & DMEMW & !LBE1n);
!CAS02n	=	(!CAS0n & DMEMR) # (!CAS0n & DMEMW & !LBE2n);
!CAS03n	=	(!CAS0n & DMEMR) # (!CAS0n & DMEMW & !LBE3n);
!CAS04n	=	(!CAS0n & DMEMR) # (!CAS0n & DMEMW & !LBE4n);
!CAS05n	=	(!CAS0n & DMEMR) # (!CAS0n & DMEMW & !LBE5n);
!CAS06n	=	(!CAS0n & DMEMR) # (!CAS0n & DMEMW & !LBE6n);
!CAS07n	=	(!CAS0n & DMEMR) # (!CAS0n & DMEMW & !LBE7n);

end PLD6;

3

241562-41

module PLD7

title 'DRAM CONTROLLER - PLD 7
INTEL CORPORATION, August 1992'

" CAS Decode PLD for Bank 1"

PLD7 device 'E224C';

x, c, z = .X., .C., .Z.;

" Inputs

CAS1n	pin	27;	"Column Address Strobe for Bank 1"
DMEMR	pin	2;	"DRAM Read Cycle"
DMEMW	pin	3;	"DRAM Write Cycle"
LBE0n	pin	4;	"Latched Byte Enables BE0# - BE7#"
LBE1n	pin	5;	
LBE2n	pin	6;	
LBE3n	pin	7;	
LBE4n	pin	9;	
LBE5n	pin	10;	
LBE6n	pin	11;	
LBE7n	pin	12;	

" Outputs

CAS10n	pin	18;	"CAS Decode outputs for Bank 1"
CAS11n	pin	19;	"
CAS12n	pin	20;	"
CAS13n	pin	21;	"
CAS14n	pin	23;	"
CAS15n	pin	24;	"
CAS16n	pin	25;	"
CAS17n	pin	26;	"

equations

ICAS10n = (ICAS1n & DMEMR) # (ICAS1n & DMEMW & !LBE0n);

ICAS11n = (ICAS1n & DMEMR) # (ICAS1n & DMEMW & !LBE1n);

ICAS12n = (ICAS1n & DMEMR) # (ICAS1n & DMEMW & !LBE2n);

ICAS13n = (ICAS1n & DMEMR) # (ICAS1n & DMEMW & !LBE3n);

ICAS14n = (ICAS1n & DMEMR) # (ICAS1n & DMEMW & !LBE4n);

ICAS15n = (ICAS1n & DMEMR) # (ICAS1n & DMEMW & !LBE5n);

ICAS16n = (ICAS1n & DMEMR) # (ICAS1n & DMEMW & !LBE6n);

ICAS17n = (ICAS1n & DMEMR) # (ICAS1n & DMEMW & !LBE7n);

end PLD7;

241562-42

module PLD8

title 'DRAM CONTROLLER - PLD 8
INTEL CORPORATION, August 1992'

" Generates Burst Ready Signals for CPU Read and Write Cycles"

PLD8 device 'E224C';

x, c, z = .X., .C., .Z.;

" Inputs

CLK	pin	2;	"CPU input clock"
CIPn	pin	16;	"DRAM Cycle in Progress"
DRAMCSn	pin	3;	"DRAM Chip Select"
WRn	pin	4;	"Write/Read Indicator"
CAS0n	pin	5;	"CAS# for Bank 0"
CAS1n	pin	6;	"CAS# for Bank 1"
BCNT	pin	7;	"Burst Count Active with 4th BRDY for Burst Cycles"
HITn	pin	17;	"DRAM Page Hit Signal"
CYC4X	pin	9;	"CPU four transfer cycle"
HLDA	pin	10;	"CPU Hold Acknowledge Signal"
RESET	pin	11;	"CPU Reset Signal"
RASn	pin	12;	"RAS# Signal"
CT	pin	13;	"CPU Cycle Track Signal"
ALE	pin	27;	"Address Latch Enable Signal"

" Outputs

MRBRDYn	pin	18;	"Burst Ready for CPU Read Cycles"
MWBRDYn	pin	19;	"Burst Ready for CPU Write Cycles"
Var1	pin	20	istype 'buffer'; "State Variable"
Var2	pin	21	istype 'buffer'; "State Variable"
Var3	pin	23	istype 'buffer'; "State Variable"
Var4	pin	24	istype 'buffer'; "State Variable"
Var5	pin	25	istype 'buffer'; "State Variable"
Var6	pin	26	istype 'buffer'; "State Variable"

" State Register Assignments

sreg1	=	[Var1, Var2, Var3];
S0	=	[0, 0, 0];
S1	=	[0, 1, 0];
S2	=	[1, 1, 0];
S3	=	[0, 1, 1];
S4	=	[1, 1, 1];
sreg2	=	[Var4, Var5, Var6];
S8	=	[0, 0, 0];
S9	=	[0, 1, 0];
S10	=	[1, 1, 0];
S11	=	[1, 1, 1];
S12	=	[1, 0, 0];
S13	=	[1, 0, 1];
S14	=	[0, 0, 1];
S15	=	[0, 1, 1];

state_diagram sreg1

```

state S0 :
  if RESET then S0 with MRBRDYn := 1; endwith else
    if (!CIPn & !DRAMCSn & !WRn & !CAS0n & !HLDA & !ALE)
      # (!CIPn & !DRAMCSn & !WRn & !CAS1n & !HLDA & !ALE) then S1
        with MRBRDYn := 1; endwith
    else S0 with MRBRDYn := 1; endwith;

state S1 :
  if RESET then S0 with MRBRDYn := 1; endwith else
    if !CYC4X then S2 with MRBRDYn := 0; endwith else
    if CYC4X then S3 with MRBRDYn := 0; endwith
    else S1 with MRBRDYn := 1; endwith;

state S2 :
  goto S0 with MRBRDYn := 1; endwith;

state S3 :
  if RESET # BCNT then S0 with MRBRDYn := 1; endwith else
    if !BCNT then S4 with MRBRDYn := 1; endwith
    else S3 with MRBRDYn := 0; endwith;

state S4 :
  if RESET # BCNT then S0 with MRBRDYn := 1; endwith else
    if !BCNT then S3 with MRBRDYn := 0; endwith
    else S4 with MRBRDYn := 1; endwith;

```

state_diagram sreg2

```

state S8 :
  if RESET then S8 with MWBRDYn := 1; endwith else
    if (!CIPn & !DRAMCSn & WRn & !HLDA & CYC4X & !ALE) then S12
      with MWBRDYn := 1; endwith else
    if (!CIPn & !DRAMCSn & WRn & !HLDA & HITn & !ALE)
      # (!CIPn & !DRAMCSn & WRn & !HLDA & RASn & !ALE) then S9
        with MWBRDYn := 1; endwith else
    if (!CIPn & !DRAMCSn & WRn & !HITn & !RASn & !HLDA & !ALE) then S10
      with MWBRDYn := 0; endwith
    else S8 with MWBRDYn := 1; endwith;

state S9 :
  if RESET then S8 with MWBRDYn := 1; endwith else
    if !IRASn then S10 with MWBRDYn := 0; endwith
    else S9 with MWBRDYn := 1; endwith;

state S10 :
  if RESET # (!CT & !HITn) then S8 with MWBRDYn := 1; endwith else
    if CT & !HITn then S11 with MWBRDYn := 1; endwith else
    if HITn then S9 with MWBRDYn := 1; endwith
    else S10 with MWBRDYn := 0; endwith;

state S11 :
  if RESET then S8 with MWBRDYn := 1; endwith
  else S9 with MWBRDYn := 1; endwith;

state S12 :
  if RESET then S8 with MWBRDYn := 1; endwith

```

else S13 with MWBRDYn := 0; endwith;

state S13 :

if RESET # BCNT then S8 with MWBRDYn := 1; endwith else
if IBCNT then S14 with MWBRDYn := 1; endwith;

state S14 :

if RESET # BCNT then S8 with MWBRDYn := 1; endwith
else if IBCNT then S13 with MWBRDYn := 0; endwith;

state S15 :

goto S8 with MWBRDYn := 1; endwith;

equations

[Var1, MRBRDYn, MWBRDYn, Var2, Var3, Var4, Var5, Var6].clk = CLK;

end PLD8;

241562-45

3

module PLD9

title 'DRAM CONTROLLER - PLD 9
INTEL CORPORATION, August 1992'

" DRAM Page Hit comparison and Row Decode PLD"

PLD9 device 'E0320';

x, c, z = .X., .C., .Z.;

" Inputs

A21	pin	1;	"CPU Address Line A21"
A22	pin	2;	"CPU Address Line A22"
A23	pin	3;	"CPU Address Line A23"
PA21	pin	4;	"Previous Address Line A21"
PA22	pin	5;	"Previous Address Line A22"
PA23	pin	6;	"Previous Address Line A23"
PROWAn	pin	7;	"Previous DRAM ROW A decode"
PROWBn	pin	8;	"Previous DRAM ROW B decode"
PHIMEMn	pin	9;	"Previous DRAM High/Low Memory Bank Decode"
MSIZA	pin	11;	"MSIZ = 0 for 256Kx36 module, MSIZ = 1 for 512Kx36 module"
MSIZB	pin	12;	"MSIZA for Row A, MSIZB for Row B"
HIT1n	pin	13;	"Address Comparison of A13-A20 with PA13-PA20"

" Outputs

HIT2n	pin	14;	"Active if A21, 22, 23 match previous PA21, 22, 23"
HIT3n	pin	15;	"Active if previous Row matches current Row and Bank"
HITn	pin	16;	"DRAM Page Hit Signal"
HIMEMn	pin	17;	"Indicates access to upper half of 512K byte"
ROWAn	pin	18;	"DRAM Row A decode"
ROWBn	pin	19;	"DRAM Row B decode"

equations

!ROWAn = (!A23 & !A22) # (MSIZA & !A23 & A22);

!ROWBn = (!MSIZA & !A23 & A22) # (MSIZA & A23 & !A22) # (MSIZB & A23 & A22);

!HIMEMn = (MSIZA & !A23 & A22) # (MSIZB & A23 & A22);

HIT2n = (!A21 & PA21) # (A21 & !PA21) # (!A22 & PA22) # (A22 & !PA22)
(!A23 & PA23) # (A23 & !PA23);

HIT3n = (!(!A23 & A22 & !MSIZA & !MSIZB & PHIMEMn & PROWAn & !PROWBn)
(!A23 & A22 & MSIZA & !MSIZB & !PHIMEMn & !PROWAn & PROWBn));

HITn = (HIT1n) # (HIT2n) # (HIT3n);

end PLD9;

241562-46

module PLD10

title 'DRAM CONTROLLER - PLD 10
INTEL CORPORATION, August 1992'

" Data Transceiver Enable for Bank 0 and Bank 1"

PLD10 device 'E224C';

x, c, z = .X., .C., .Z.;

" Inputs

CLK	pin	2;	"CPU input clock"
CIPn	pin	12;	"DRAM Cycle in Progress"
DRAMCSn	pin	3;	"DRAM Chip Select"
BANKSEL	pin	4;	"BANKSEL indicates Bank 0 or Bank 1 access"
MRBRDYN	pin	5;	"BRDY for Read cycles"
WRn	pin	6;	"Write/Read Indicator"
CYC4X	pin	7;	"CPU four transfer cycle"
HLDA	pin	10;	"CPU Hold Acknowledge signal"
RESET	pin	9;	"CPU Reset Signal"
RASn	pin	13;	"DRAM RAS# signal"
CT	pin	16;	"CPU Cycle Track signal"

" Outputs

DEN0A	pin	18;	"Data XCVR Enable Bank 0 (single rd/wr, burst rd/wr bank 0)"
DEN1A	pin	26;	"Data XCVR Enable Bank 1 (burst read bank 1, BANKSEL = 0)"
Var1	pin	19;	"State Variable"
Var2	pin	20;	"State Variable"
DEN1C	pin	21;	"Data XCVR Enable Bank 1 (burst writes)"
DEN0B	pin	23;	"Data XCVR Enable Bank 0 (burst read bank 0, BANKSEL = 1)"
DEN1B	pin	24;	"Data XCVR Enable Bank 1 (single rd/wr, burst read bank 1)"
DEN	pin	25;	

" State Register Assignments

```
sreg = [Var1, Var2, DEN];
S0 = [0, 0, 0];
S1 = [0, 0, 1];
S2 = [0, 1, 1];
S3 = [0, 1, 0];
S4 = [1, 1, 1];
S5 = [1, 1, 0];
```

DRAMCYCn = CIPn # DRAMCSn # RASn;

state_diagram sreg

```
state S0 :
    if RESET then S0 else
    if (!DRAMCYCn & !CT & !CYC4X & !HLDA)
        # (!DRAMCSn & !CIPn & !CT & CYC4X & !HLDA & WRn)
        # (!DRAMCYCn & HLDA) then S1 else
    if (!DRAMCYCn & !CT & CYC4X & !HLDA & !WRn) then S2
    else S0;

state S1 :
    if RESET # CIPn # (!MRBRDYN & !HLDA) then S0
```

241562-51


```

else S1;

state S2 :    if RESET then S0 else
               if !MRBRDYn then S3
               else S2;

state S3 :    if RESET then S0 else
               if !MRBRDYn then S4
               else S3;

state S4 :    if RESET then S0 else
               if !MRBRDYn then S5
               else S4;

state S5 :    if RESET # !MRBRDYn then S0
               else S5;

equations

[Var1, Var2, DEN].clk = CLK;

DEN0A = !DEN.FB;
DEN1A = !DEN0A;
DEN0B = !DEN1B;
DEN1B = !DEN.FB;
DEN1C = !DEN.FB;
DEN0A.OE = !BANKSEL;
DEN1A.OE = CYC4X & IHLDA & !BANKSEL & DEN0A & !WRn & !DRAMCYCn;
DEN0B.OE = CYC4X & IHLDA & BANKSEL & DEN1B & !WRn & !DRAMCYCn;
DEN1B.OE = BANKSEL;
DEN1C.OE = CYC4X & IHLDA & WRn & !BANKSEL;

end PLD10;

```

241562-52

module PLD11

title 'DRAM CONTROLLER - PLD 11
INTEL CORPORATION, August 1992'

" Activates Write Data Register for Bank 0 and Bank 1"

PLD11 device 'E0320';

x, c, z = .X., .C., .Z.;

" Inputs

CLK	pin	1;	"CPU input clock
CIPn	pin	2;	"DRAM Cycle in Progress
PIPECYCn	pin	3;	"Active during pipelined cycles
BANKSEL	pin	4;	"Indicates Bank 0 or Bank 1 access
MWBRDYn	pin	5;	"BRDY for Write Cycles
CYC4X	pin	6;	"CPU four transfer cycle
HLDA	pin	7;	"CPU Hold Acknowledge output
RESET	pin	8;	"CPU Reset Input
CAS0n	pin	9;	"CAS# for Bank 0
CAS1n	pin	11;	"CAS# for Bank 1
DMEMW	pin	12;	"DRAM write cycle
LA3	pin	19;	"Latched Address line 3 signal

" Outputs

DCP0	pin	13;	"Activates Write Data Register for Bank 0"
DCP1	pin	14;	"Activates Write Data Register for Bank 1"
Var1	pin	15;	"State Variable"
Var2	pin	16;	"State Variable"
Var3	pin	17;	"State Variable"
Var4	pin	18;	"State Variable"

" State Register Assignments

sreg1	=	[DCP0, Var1, Var2];
S00	=	[0, 0, 0];
S01	=	[1, 0, 0];
S02	=	[1, 0, 1];
S03	=	[0, 0, 1];
S04	=	[1, 1, 1];
S05	=	[0, 1, 1];

sreg2	=	[DCP1, Var3, Var4];
S10	=	[0, 0, 0];
S11	=	[1, 0, 0];
S12	=	[0, 1, 0];
S13	=	[1, 1, 0];
S14	=	[0, 1, 1];
S15	=	[0, 0, 1];
S16	=	[1, 0, 1];

state_diagram sreg1

state S00 : if RESET then S00 else
if (ICIPn & PIPECYCn & !LA3 & DMEMW & !CYC4X

3

241562-53

```

        & !HLDA) # (!CIPn & !LA3 & DMEW
        & HLDA) then S01 else
    if (!CIPn & !BANKSEL & DMEW & CYC4X & !HLDA & PIPECYCn)
    then S02
    else S00;

state S01 :    if RESET # !CAS0n then S00
               else S01;

state S02 :    if RESET then S00 else
               if (!CAS0n & CYC4X) then S03 else
               if (!CAS0n & !CYC4X) then S00
               else S02;

state S03 :    if RESET then S00
               else S04;

state S04 :    if RESET # !CAS0n then S05
               else S04;

state S05 :    goto S00;

state_diagram sreg2

state S10 :    if RESET then S10 else
               if (!CIPn & PIPECYCn & LA3 & DMEW & !CYC4X
               & !HLDA) # (!CIPn & LA3 & DMEW
               & HLDA) then S11 else
               if (!CIPn & !MWBRDYn & DMEW & CYC4X & !HLDA) then S12
               else S10;

state S11 :    if RESET # !CAS1n then S10
               else S11;

state S12 :    if RESET # !CYC4X then S10
               else S13;

state S13 :    if RESET then S10
               else if !CAS1n then S14
               else S13;

state S14 :    if RESET then S10
               else S15;

state S15 :    if RESET then S10
               else S16;

state S16 :    if RESET # CAS1n then S10
               else S16;

equations
    [Var1, Var2, Var3, Var4, DCP0, DCP1].clk = CLK;

end PLD11;

```

241562-54

module PLD12

title 'DRAM CONTROLLER - PLD 12
INTEL CORPORATION, August 1992'

- " This PLD generates Bank Select during CPU Burst Cycles,"
- " Burst Count during Burst Read and Writeback Cycles,"
- " and Row Address Latch Enable for Page Comparison"

PLD12 device 'E0320';

x, c, z = .X., .C., .Z.;

Inputs

CLK	pin	1;	"CPU Clock Input
CIPn	pin	2;	"DRAM Cycle in Progress
DRAMCSn	pin	3;	"DRAM Chip Select
LA3	pin	4;	"Latched Address line A3
MRBRDYn	pin	5;	"BRDY# for Read Cycles
MWBRDYn	pin	6;	"BRDY# for Write Cycles
CYC4X	pin	7;	"CPU four transfer cycle
HLDA	pin	8;	"CPU HOLD Acknowledge Output
RESET	pin	9;	"CPU RESET signal
HITn	pin	11;	"DRAM Page Hit Signal

Outputs

BANKSEL	pin	13;	"Bank Select during CPU burst cycles"
Var1	pin	14;	"State Variable"
BCNT	pin	15;	"Burst Count active with 4th BRDY"
Var2	pin	16;	"State Variable"
Var3	pin	17;	"State Variable"
Var4	pin	18;	"State Variable"
RALE	pin	19;	"Row Address Latch Enable"
BCNT3	pin	12;	"active with 3rd BRDY"

State Register Assignments

sreg1	=	[BANKSEL, Var1];
S00	=	[0, 0];
S01	=	[1, 0];
S02	=	[0, 1];
S03	=	[1, 1];
sreg2	=	[Var2, Var3, Var4, BCNT];
S10	=	[0, 0, 0, 0];
S11	=	[0, 0, 1, 0];
S12	=	[0, 1, 1, 0];
S13	=	[0, 1, 0, 0];
S14	=	[1, 1, 0, 0];
S15	=	[1, 1, 1, 1];

state_diagram sreg1

```
state S00 :
    if RESET then S00 else
    if (CIPn & IDRAMCSn & LA3 & IHLDA) then S01 else
```



```

        if (!ICIPn & !DRAMCSn & !LA3 & !HLDA) then S02
        else S00;

state S01 :
    if RESET then S00 else
    if (CYC4X & BCNT.FB & !MRBRDn) # (CYC4X & BCNT.FB & !MWBRDn)
        # (!CYC4X & !MRBRDn) # (!CYC4X & !MWBRDn) then S03
    else S01;

state S02 :
    if RESET # (CYC4X & BCNT.FB & !MRBRDn) #
        (CYC4X & BCNT.FB & !MWBRDn) # (!CYC4X & !MRBRDn) #
        (!CYC4X & !MWBRDn) then S00
    else S02;

state S03 :
    if RESET then S00 else
    if (!ICIPn & !DRAMCSn & LA3 & !HLDA) then S01
    else S00;

state_diagram sreg2

state S10 :
    if RESET then S10 with BCNT3 := 0; endwith else
    if (!ICIPn & !DRAMCSn & CYC4X & !HLDA) then S11 with BCNT3 := 0; endwith
    else S10 with BCNT3 := 0; endwith;

state S11 :
    if RESET # !CYC4X then S10 with BCNT3 := 0; endwith else
    if !MRBRDn # !MWBRDn then S12 with BCNT3 := 0; endwith
    else S11 with BCNT3 := 0; endwith;

state S12 :
    if RESET then S10 with BCNT3 := 0; endwith else
    if !MRBRDn # !MWBRDn then S13 with BCNT3 := 0; endwith
    else S12 with BCNT3 := 0; endwith;

state S13 :
    if RESET then S10 with BCNT3 := 0; endwith else
    if !MRBRDn # !MWBRDn then S14 with BCNT3 := 1; endwith
    else S13 with BCNT3 := 0; endwith;

state S14 :
    if RESET then S10 with BCNT3 := 0; endwith
    else S15 with BCNT3 := 0; endwith;

state S15 :
    goto S10 with BCNT3 := 0; endwith;

state_diagram [RALE]

state [0] :
    if RESET then [0] else
    if !ICIPn & !DRAMCSn & !HITn then [1]
    else [0];

state [1] :
    if RESET # !HITn then [0]

```

else [1];

equations

[Var1, BCNT3, Var2, Var3, Var4, BANKSEL, BCNT, RALE].clk = CLK;

BANKSEL.OE = !HLDA;

end PLD12;

241562-57

3

module PLD13

title 'DRAM CONTROLLER - PLD 13
INTEL CORPORATION, August 1992'

" This PLD Generates the address latch enable for cycle pipelining"

PLD13 device 'P22V10C';

x, c, z = .X., .C., .Z.;

" Inputs

CLK	pin	2;	"CPU Input Clock
CIPn	pin	27;	"DRAM Cycle in Progress
DRAMCSn	pin	3;	"DRAM Chip Select
ADSn	pin	4;	"CPU Address Status Output
MRBRDYn	pin	5;	"BRDY# for Read Cycles
MWBRDYn	pin	6;	"BRDY# for Write Cycles
CYC4X	pin	7;	"CPU Four Transfer Cycle
HLDA	pin	12;	"CPU Hold Acknowledge Signal
RESET	pin	9;	"CPU Reset Input
WRn	pin	10;	"Write/Read Indicator
NAAn	pin	11;	"CPU Next Address Input
BCNT	pin	13;	"Burst Count Active with 4th BRDY for Burst Cycles
MSBURSTn	pin	26;	"EISA Burst Cycle Indicator
DBCLK	pin	25;	"Delayed BCLK
CMDn	pin	16;	"EISA Cycle Command Indicator
BCNT3	pin	23;	"Burst Count Active with 3rd BRDY for Burst Cycles

" Outputs

!ALE	pin	20	istype 'invert';
Var1	pin	18	istype 'buffer'; "State Variable"
Var2	pin	19	istype 'buffer'; "State Variable"

" State Register Assignments

sreg1	=	[ALE, Var1, Var2];
S0	=	[0, 0, 0]; "ALE = 1;
S1	=	[1, 0, 0]; "ALE = 0;
S2	=	[1, 0, 1]; "ALE = 0;
S3	=	[0, 1, 1]; "ALE = 1;
S4	=	[1, 1, 0]; "ALE = 0;
S5	=	[0, 1, 0]; "ALE = 1;

state_diagram sreg1

```

state S0 :
  if RESET then S0 else
    if (!CIPn & !DRAMCSn) # (!ADSn & !HLDA) then S1
    else S0;

state S1 :
  if RESET # CIPn # DRAMCSn # (CYC4X & BCNT & WRn & !HLDA)
    # (MSBURSTn & CIPn & HLDA) then S0 else
  if !NAA & !HLDA then S2 else
  if !DBCLK & !MSBURSTn & HLDA & !CMDn then S3

```

241562-58

else S1;

state S2 :

if RESET # (ICYC4X & !MRBRDYn & ADSn) # CIPn then S0 else
if (ICYC4X & !MRBRDYn & !ADSn) then S3 else
if !MWBRDYn then S4 else
if (BCNT3 & !WRn & CYC4X) then S5
else S2;

state S3 :

if RESET # CIPn # DRAMCSn # (MSBURSTn & DBCLK) then S0 else
if (!CIPn & !DRAMCSn) # (DBCLK & !MSBURSTn & HLDA & !CMDn) then S1
else S3;

state S4 :

goto S0;

state S5 :

if RESET then S0
else S3;

equations

[ALE, Var1, Var2].clk = CLK;

[ALE, Var1, Var2].AR = RESET;

end PLD13;

3

241562-59

module PLD14

title 'DRAM CONTROLLER - PLD 14
INTEL CORPORATION, August 1992'

" Burst Address Generation PLD - This PLD generates the Burst Address and supplies the lowest address line to the DRAM array.

PLD14 device 'E224C';

x, c, z = .X., .C., .Z.;

" Inputs

CLK	pin	2;	"CPU Input Clock
CIPn	pin	27;	"DRAM Cycle in Progress
DRAMCSn	pin	3;	"DRAM Chip Select
LA4	pin	4;	"Latched Address line A4
LA13	pin	5;	"Latched Address line A13
MRBRDYn	pin	7;	"BRDY# for Read Cycles
MWBRDYn	pin	12;	"BRDY# for Write Cycles
CYC4X	pin	9;	"CPU Four Transfer Cycle
HLDA	pin	10;	"CPU Hold Acknowledge Output
RESET	pin	11;	"CPU Reset input
RASn	pin	13;	"DRAM RAS# signal
BCNT	pin	16;	"Burst Count Active with 4th BRDY# for Burst Cycles
ALE	pin	17;	"Address Latch Enable signal

" Outputs

BOAMA0	pin	24;	"A0 for Bank 0, Row A
BOBMA0	pin	25;	"A0 for Bank 0, Row B
B1AMA0	pin	26;	"A0 for Bank 1, Row A
B1BMA0	pin	18;	"A0 for Bank 1, Row B
BA	pin	19;	"
Var3	pin	20	istype 'buffer'; "State Variable
Var1	pin	21	istype 'buffer'; "State Variable
Var2	pin	23	istype 'buffer'; "State Variable

" State Register Assignments

sreg	=	[Var1, Var2, Var3];
S0	=	[0, 0, 0];
S1	=	[0, 0, 1];
S2	=	[0, 1, 0];
S3	=	[0, 1, 1];
S4	=	[1, 1, 0];
S5	=	[1, 1, 1];
S6	=	[1, 0, 1];

state_diagram sreg

state S0 :

```

if RESET then S0 with BA := 1; endwith else
if (!CIPn & !DRAMCSn & !LA4 & !HLDA & !ALE) then S1 with BA := 0; endwith else
if (!CIPn & !DRAMCSn & LA4 & !HLDA & !ALE) then S2 with BA := 1; endwith
else S0 with BA := 1; endwith;

```

241562-60

state S1 :

```
if RESET then S0 with BA := 1; endwith else
if (!MRBRDYN # !MWBRDYN) & CYC4X then S3 with BA := 0; endwith else
if !MRBRDYN & !CYC4X then S0 with BA := 1; endwith else
if !MWBRDYN & !CYC4X then S5 with BA := 0; endwith
else S1 with BA := 0; endwith;
```

state S2 :

```
if RESET then S0 with BA := 1; endwith else
if (!MRBRDYN # !MWBRDYN) & CYC4X then S4 with BA := 1; endwith else
if !MRBRDYN & !CYC4X then S0 with BA := 1; endwith else
if !MWBRDYN & !CYC4X then S6 with BA := 1; endwith
else S2 with BA := 1; endwith;
```

state S3 :

```
if RESET then S0 with BA := 1; endwith else
if (!MRBRDYN # !MWBRDYN) & IBCNT then S2 with BA := 1; endwith else
if BCNT then S0 with BA := 1; endwith
else S3 with BA := 0; endwith;
```

state S4 :

```
if RESET then S0 with BA := 1; endwith else
if (!MRBRDYN # !MWBRDYN) & IBCNT then S1 with BA := 0; endwith else
if BCNT then S0 with BA := 1; endwith
else S4 with BA := 1; endwith;
```

state S5 :

```
if RESET # ALE then S0 with BA := 1; endwith
else S5 with BA := 0; endwith;
```

state S6 :

```
if RESET # ALE then S0 with BA := 1; endwith
else S6 with BA := 1; endwith;
```

equations

[Var1, BA, Var2, Var3].clk = CLK;

B0AMA0 = (BA & !IRASn & !HLDA) # (LA13 & RASn) # (LA4 & !IRASn & HLDA);

B0BMA0 = (BA & !IRASn & !HLDA) # (LA13 & RASn) # (LA4 & !IRASn & HLDA);

B1AMA0 = (BA & !IRASn & !HLDA) # (LA13 & RASn) # (LA4 & !IRASn & HLDA);

B1BMA0 = (BA & !IRASn & !HLDA) # (LA13 & RASn) # (LA4 & !IRASn & HLDA);

end PLD14;

241562-61

module PLD15

title 'DRAM CONTROLLER - PLD 15
INTEL CORPORATION, August 1992'

" Generate NA# input to the CPU for pipelining"

PLD15 device 'E224C';

x, c, z = .X., .C., .Z.;

" Inputs

CLK	pin	2;	"CPU input clock"
ADSn	pin	3;	"CPU ADS# Output"
CIPn	pin	4;	"DRAM Cycle in Progress"
PIPECYCn	pin	5;	"Active for Pipelined Cycles"
DRAMCSn	pin	6;	"DRAM Chip Select"
BCNT	pin	7;	"Burst Count Active with 4th BRDY for Burst Cycles"
WRn	pin	9;	"CPU Write/Read Indication"
CACHEn	pin	10;	"CPU CACHE# output"
HITn	pin	11;	"DRAM Page Hit Indicator"
RESET	pin	12;	"CPU Reset Signal"
HLDA	pin	13;	"CPU Hold Acknowledge Signal"
CT	pin	16;	"Cycle Track Output"
MIOn	pin	17;	"Memory/IO Indicator"
P5BRDYn	pin	27;	"BRDY# input to the CPU"

" Outputs

NAn	pin	19;	"Next Address Input for CPU"
Var1	pin	20;	"State Variable"
Var2	pin	21;	"State Variable"
Var3	pin	18;	"State Variable"
CPUIOR	pin	23;	"I/O Read generated by CPU"
CPUIOWR	pin	24;	"I/O Write generated by CPU"

" State Register Assignments

```
sreg = [Var1, Var2, Var3];
S0 = [0, 0, 0];
S1 = [0, 0, 1];
S2 = [0, 1, 1];
S3 = [0, 1, 0];
S4 = [1, 1, 1];
```

state_diagram sreg

state S0 :

```
if RESET then S0 with NAn := 1; endwith else
if !ADSn & WRn & !PIPECYCn & !HLDA then S3 with NAn := 1; endwith else
if !ADSn & WRn & CACHEn & !HITn & !HLDA then S2 with NAn := 0; endwith else
if (!ADSn & !WRn & PIPECYCn & !HLDA)
# (!ADSn & WRn & HITn & CACHEn) then S1 with NAn := 1; endwith else
if (!ADSn & !WRn & !PIPECYCn & !HLDA) then S4 with NAn := 1; endwith
else S0 with NAn := 1; endwith;
```

state S1 :

241562-62

```

if RESET then S0 with NAn := 1; endwith else
if (ICIPn & !DRAMCSn & !WRn & !HITn) # (WRn & CACHEn & !HITn & !CT)
# (ICIPn & !DRAMCSn & !WRn & HITn & !P5BRDYn) then S2
with NAn := 0; endwith
else S1 with NAn := 1; endwith;

state S2 :
goto S0 with NAn := 1; endwith;

state S3 :
if RESET then S0 with NAn := 1; endwith else
if BCNT then S1 with NAn := 1; endwith
else S3 with NAn := 1; endwith;

state S4 :
if RESET then S0 with NAn := 1; endwith else
if (PIPECYCn & ICIPn & !DRAMCSn & !WRn & !HITn) then S2 with NAn := 0; endwith
else S4 with NAn := 1; endwith;

```

equations

[Var1, NAn, Var2, Var3, CPUIORD, CPUIOWR].clk = CLK;

CPUIORD := (!ADSn & !MION & !WRn & !HLDA) # (CPUIORD.FB & P5BRDYn);

CPUIOWR := (!ADSn & !MION & WRn & !HLDA) # (CPUIOWR.FB & P5BRDYn);

end PLD15;

3

241562-63

module PLD16

title 'DRAM CONTROLLER - PLD 16
INTEL CORPORATION, August 1992'

" Buffers and generates Byte Enables between CPU Memory Bus and EBC Host bus"

PLD16 device 'E0600C'; "Implemented with Intel 85C060 PLD"

x, c, z = .X., .C., .Z.;

" Inputs

ISAMEMW	pin	3;	"ISA DRAM Write Cycle
EISAMEMW	pin	13;	"EISA DRAM Write Cycle
CPUIORD	pin	17;	"CPU I/O Read Cycle
HITMn	pin	27;	"CPU Hit to a Modified Line
HLDA	pin	4;	"CPU Hold Acknowledge Output

" Input/Outputs

HBE3n	pin	5;	"Host Bus Byte Enables 0-3 for 32-bit Bus
HBE2n	pin	6;	
HBE1n	pin	7;	
HBE0n	pin	8;	
HA2	pin	9;	"Address Line 2 for 32-bit bus
BE7n	pin	26;	"CPU Byte Enable 0-7 for 64-bit Bus
BE6n	pin	25;	
BE5n	pin	24;	
BE4n	pin	23;	
BE3n	pin	22;	
BE2n	pin	21;	
BE1n	pin	20;	
BE0n	pin	18;	

" Outputs

HDOEn	pin	12;	"Controls Output Enables on CPU bus side of Data Buffers "(EBBs) between 64- & 32-bit bus.
HAOEn	pin	10;	"Controls Output Enable of Address Buffers between CPU bus "and 32-bit Host Bus

equations

!BE7n	=	HA2 # !HBE3n;
BE7n.OE	=	HLDA & HITMn;
!BE6n	=	HA2 # !HBE2n;
BE6n.OE	=	HLDA & HITMn;
!BE5n	=	HA2 # !HBE1n;
BE5n.OE	=	HLDA & HITMn;
!BE4n	=	HA2 # !HBE0n;
BE4n.OE	=	HLDA & HITMn;
!BE3n	=	!HA2 # !HBE3n;
BE3n.OE	=	HLDA & HITMn;

IBE2n = IHA2 # IBE2n;
 BE2n.OE = HLDA & HITMn;

 IBE1n = IHA2 # IBE1n;
 BE1n.OE = HLDA & HITMn;

 IBE0n = IHA2 # IBE0n;
 BE0n.OE = HLDA & HITMn;

 HA2 = IBE7n # IBE6n # IBE5n # IBE4n;
 HA2.OE = IHLDA;

 IHE3n = (HA2 & IBE7n) # (IHA2 & IBE3n);
 HBE3n.OE = IHLDA;

 IHE2n = (HA2 & IBE6n) # (IHA2 & IBE2n);
 HBE3n.OE = IHLDA;

 IHE1n = (HA2 & IBE5n) # (IHA2 & IBE1n);
 HBE3n.OE = IHLDA;

 IHE0n = (HA2 & IBE4n) # (IHA2 & IBE0n);
 HBE3n.OE = IHLDA;

 IHDOEn = (ISAMEMW & IHAOEn) # (EISAMEMW & IHAOEn)
 # (CPUIORD & IHAOEn);

 IHAOEn = HITMn;

 end PLD16;

module PLD17

title 'DRAM CONTROLLER - PLD 17
INTEL CORPORATION, August 1992'

* ISA State Tracker - This PLD tracks ISA cycles to DRAM*

PLD17 device 'E0320';

x, c, z = .X., .C., .Z.;

* Inputs

CLK	pin	1;	*CPU Input Clock
SEMSTR16n	pin	2;	*Signal to indicate 16-bit ISA master (synchronized)
SMRDCn	pin	3;	*ISA Memory Read Control Strobe (synchronized)
SMWTCn	pin	4;	*ISA Memory Write Control Strobe (synchronized)
HLDA	pin	5;	*CPU Hold Acknowledge Output
HITMn	pin	6;	*CPU Hit to a Modified Line Signal
DRAMCSn	pin	7;	*DRAM Chip Select
DRAMDISn	pin	8;	*DRAM Disable Signal
RESET	pin	9;	*CPU Reset Signal

* Outputs

ISAMEMR	pin	12;	*ISA Read to DRAM
ISAMEMW	pin	13;	*ISA Write to DRAM
CIPn	pin	14;	*DRAM Cycle in Progress (ISA Cycles)
NCHRDYn	pin	15;	*CHRDY active to Hold off ISA bus master
EADSRIn	pin	16;	*EADS to CPU for ISA Read Cycles
Var1	pin	17;	*State Variable
Var2	pin	18;	*State Variable
Var3	pin	19;	*State Variable

* state register assignments

```
sreg1 = [Var1, Var2, Var3];
S0 = [0, 0, 0];
S1 = [0, 0, 1];
S2 = [0, 1, 1];
S3 = [0, 1, 0];
S4 = [1, 1, 0];
```

```
ISARD = !DRAMCSn & !DRAMDISn & !SMRDCn & HLDA & !SEMSTR16n;
```

```
ISAWR = !DRAMCSn & !SMWTCn & HLDA & !SEMSTR16n;
```

state_diagram sreg1

state S0 :

```
if RESET then S0 with CIPn := 1; NCHRDYn := 1; EADSRIn := 1; endwith else
if (ISARD # ISAWR) then S1 with CIPn := 1; NCHRDYn := 0; EADSRIn := 0; endwith
else S0 with CIPn := 1; NCHRDYn := 1; EADSRIn := 1; endwith;
```

state S1 :

```
if RESET then S0 with CIPn := 1; NCHRDYn := 1; EADSRIn := 1; endwith
else S2 with CIPn := 1; NCHRDYn := 0; EADSRIn := 1; endwith;
```

```
state S2 :
  if RESET then S0 with CIPn := 1; NCHRDYn := 1; EADSRIn := 1; endwith
  else S3 with CIPn := 1; NCHRDYn := 0; EADSRIn := 1; endwith;
```

```
state S3 :
  if RESET then S0 with CIPn := 1; NCHRDYn := 1; EADSRIn := 1; endwith else
  if HITMn then S4 with CIPn := 0; NCHRDYn := 1; EADSRIn := 1; endwith
  else S3 with CIPn := 1; NCHRDYn := 0; EADSRIn := 1; endwith;
```

```
state S4 :
  if RESET # SEMSTR16n # (SMRDCn & SMWTCn) then S0
    with CIPn := 1; NCHRDYn := 1; EADSRIn := 1; endwith
  else S4 with CIPn := 0; NCHRDYn := 1; EADSRIn := 1; endwith;
```

equations

[Var1, CIPn, NCHRDYn, EADSRIn, Var2, Var3].clk = CLK;

CIPn.OE = HLDA & ISEMSTR16n & HITMn;

ISAMEMR = ISARD;

ISAMEMW = ISAWR;

end PLD17;

3

241562-67

module PLD18

title 'DRAM CONTROLLER - PLD 18'
INTEL CORPORATION, August 1992'

" EISA State Tracker - This PLD tracks EISA cycles to DRAM"

PLD18 device 'P22V10C';

x, c, z = .X., .C., .Z.;

" Inputs

CLK	pin	2;	"CPU Clock Input
EMSTR16n	pin	17;	"Signal to indicate 16-bit ISA master
CMDn	pin	3;	"EISA Command Signal
STARTn	pin	4;	"EISA Cycle Start Signal
MSBURSTn	pin	5;	"Active for EISA Burst Cycles
DBCLK	pin	6;	"Delayed BCLK
DRAMCSn	pin	7;	"DRAM Chip Select
DRAMDISn	pin	12;	"DRAM Disable Signal
RESET	pin	9;	"Reset Signal
HLDA	pin	10;	"CPU Hold Acknowledge Signal
HITMn	pin	11;	"CPU Hit to a Modified Line Signal
REFRESHn	pin	13;	"Active for Refresh Cycles

" Outputs

CIPn	pin	18;	"DRAM Cycle in Progress (EISA cycles)
BNKSWTCH	pin	19;	"Bank Switch for EISA cycles
EADSREn	pin	20;	"EADS# to CPU for EISA Read Cycles
NEXRDYn	pin	21;	"EXRDY# active to hold off EISA bus master
Var1	pin	23	istype 'buffer'; "State Variable
Var2	pin	24	istype 'buffer'; "State Variable
Var3	pin	25	istype 'buffer'; "State Variable
Var4	pin	26	istype 'buffer'; "State Variable

" state register assignments

sreg1	=	[Var1, Var2, Var3, Var4];
S0	=	[0, 0, 0, 0];
S1	=	[0, 0, 0, 1];
S2	=	[0, 0, 1, 1];
S3	=	[0, 1, 1, 1];
S4	=	[0, 1, 1, 0];
S5	=	[1, 1, 1, 0];
S6	=	[1, 1, 0, 0];
S7	=	[1, 0, 0, 0];
S8	=	[0, 1, 0, 0];

GDRAMCS = !DRAMCSn # !REFRESHn;

state_diagram sreg1

state S0 :

if RESET then S0
with CIPn := 1; NEXRDYn := 1; EADSREn := 1; BNKSWTCH := 0; endwith else
if (DBCLK & GDRAMCS & !STARTn & HLDA & EMSTR16n & DRAMDISn) then S1

241562-68

```

        with CIPn := 1; NEXRDYn := 0; EADSREn := 0; BNKSWTCH := 0; endwith
    else S0 with CIPn := 1; NEXRDYn := 1; EADSREn := 1; BNKSWTCH := 0; endwith;

```

state S1 :

```

    if RESET then S0

```

```

        with CIPn := 1; NEXRDYn := 1; EADSREn := 1; BNKSWTCH := 0; endwith

```

```

    else S2 with CIPn := 1; NEXRDYn := 0; EADSREn := 1; BNKSWTCH := 0; endwith;

```

state S2 :

```

    if RESET then S0

```

```

        with CIPn := 1; NEXRDYn := 1; EADSREn := 1; BNKSWTCH := 0; endwith

```

```

    else S3 with CIPn := 1; NEXRDYn := 0; EADSREn := 1; BNKSWTCH := 0; endwith;

```

state S3 :

```

    if RESET then S0

```

```

        with CIPn := 1; NEXRDYn := 1; EADSREn := 1; BNKSWTCH := 0; endwith else

```

```

    if HLDA & HITMn then S6

```

```

        with CIPn := 0; NEXRDYn := 1; EADSREn := 1; BNKSWTCH := 0; endwith else

```

```

    if !HLDA & HITMn then S4

```

```

        with CIPn := 1; NEXRDYn := 0; EADSREn := 1; BNKSWTCH := 0; endwith

```

```

    else S3 with CIPn := 1; NEXRDYn := 0; EADSREn := 1; BNKSWTCH := 0; endwith;

```

state S4 :

```

    if RESET then S0

```

```

        with CIPn := 1; NEXRDYn := 1; EADSREn := 1; BNKSWTCH := 0; endwith else

```

```

    if HLDA & HITMn then S5

```

```

        with CIPn := 1; NEXRDYn := 0; EADSREn := 1; BNKSWTCH := 0; endwith

```

```

    else S4 with CIPn := 1; NEXRDYn := 0; EADSREn := 1; BNKSWTCH := 0; endwith;

```

state S5 :

```

    if RESET then S0

```

```

        with CIPn := 1; NEXRDYn := 1; EADSREn := 1; BNKSWTCH := 0; endwith

```

```

    else S6 with CIPn := 0; NEXRDYn := 1; EADSREn := 1; BNKSWTCH := 0; endwith;

```

state S6 :

```

    if RESET # (CMDn & DBCLK & STARTn) then S0

```

```

        with CIPn := 1; NEXRDYn := 1; EADSREn := 1; BNKSWTCH := 0; endwith else

```

```

    if !DBCLK & !CMDn & !MSBURSTn then S7

```

```

        with CIPn := 0; NEXRDYn := 1; EADSREn := 1; BNKSWTCH := 1; endwith

```

```

    else S6 with CIPn := 0; NEXRDYn := 1; EADSREn := 1; BNKSWTCH := 0; endwith;

```

state S7 :

```

    if RESET then S0

```

```

        with CIPn := 1; NEXRDYn := 1; EADSREn := 1; BNKSWTCH := 0; endwith else

```

```

    if DBCLK then S8

```

```

        with CIPn := 0; NEXRDYn := 1; EADSREn := 1; BNKSWTCH := 0; endwith

```

```

    else S7 with CIPn := 0; NEXRDYn := 1; EADSREn := 1; BNKSWTCH := 1; endwith;

```

state S8 :

```

    if RESET # (CMDn & DBCLK) # (MSBURSTn & !DBCLK) then S0

```

```

        with CIPn := 1; NEXRDYn := 1; EADSREn := 1; BNKSWTCH := 0; endwith else

```

```

    if !DBCLK & !MSBURSTn then S7

```

```

        with CIPn := 0; NEXRDYn := 1; EADSREn := 1; BNKSWTCH := 1; endwith

```

```

    else S8 with CIPn := 0; NEXRDYn := 1; EADSREn := 1; BNKSWTCH := 0; endwith;

```

equations

```

[Var1, CIPn, NEXRDYn, EADSREn, BNKSWTCH, Var2, Var3, Var4].clk = CLK;

```

3

[Var1, CIPn, NEXRDYn, EADSREn, BNKSWTCH, Var2, Var3, Var4].AR = RESET;

CIPn.OE = HLDA & EMSTR16n & HITMn & DRAMDISn;

end PLD18;

241562-70

module PLD19

title 'DRAM CONTROLLER - PLD 19
INTEL CORPORATION, August 1992'

" Buffers and generates Data Parity between CPU Memory Bus and EBC Host bus"

PLD19 device 'E0600C'; "Implemented with Intel 85C060 PLD"

x, c, z = .X., .C., .Z.;

" Inputs

ISAMEMR	pin	3;	"ISA DRAM Read Cycle
EISAMEMR	pin	4;	"EISA DRAM Read Cycle
CPUIOWR	pin	12;	"CPU I/O Write Cycle
HITMn	pin	13;	"CPU Hit to a Modified Line Signal
HLDA	pin	17;	"CPU Hold Acknowledge Output
HA2	pin	27;	"Address Line 2 on the 32-bit Host Bus

" Input/Outputs

HDP3	pin	5;	"Host Data Parity 7-0 for 32-bit Bus
HDP2	pin	6;	
HDP1	pin	7;	
HDP0	pin	8;	
DP7	pin	26;	"CPU Data Parity 7-0 for 64-bit Bus
DP6	pin	25;	
DP5	pin	24;	
DP4	pin	23;	
DP3	pin	22;	
DP2	pin	21;	
DP1	pin	20;	
DP0	pin	18;	

" Outputs

EDOELn	pin	10;	"Controls Output Enable for lower 32 bits on Host Bus side of "
			"data buffers (EBBs)."
EDOEHn	pin	9;	"Controls Output Enable for upper 32 bits on Host Bus side of "
			"data buffers (EBBs)."

equations

DP7	=	HA2 # DP3;
DP7.OE	=	HLDA & HITMn;
DP6	=	HA2 # DP2;
DP6.OE	=	HLDA & HITMn;
DP5	=	HA2 # DP1;
DP5.OE	=	HLDA & HITMn;
DP4	=	HA2 # DP0;
DP4.OE	=	HLDA & HITMn;
DP3	=	!HA2 # DP3;
DP3.OE	=	HLDA & HITMn;


```

DP2      =      IHA2 # DP2;
DP2.OE   =      HLDA & HITMn;

DP1      =      IHA2 # DP1;
DP1.OE   =      HLDA & HITMn;

DP0      =      IHA2 # DP0;
DP0.OE   =      HLDA & HITMn;

HDP3     =      (HA2 & DP7) # (IHA2 & DP3);
HDP3.OE  =      !HLDA;

HDP2     =      (HA2 & DP6) # (IHA2 & DP2);
HDP2.OE  =      !HLDA;

HDP1     =      (HA2 & DP5) # (IHA2 & DP1);
HDP1.OE  =      !HLDA;

HDP0     =      (HA2 & DP4) # (IHA2 & DP0);
HDP0.OE  =      !HLDA;

!EDOEHn  =      (ISAMEMR & HITMn & HA2)
                # (EISAMEMR & HITMn & HA2)
                # (CPUIOWR & HITMn & HA2);

!EDOELn  =      (ISAMEMR & HITMn & IHA2)
                # (EISAMEMR & HITMn & IHA2)
                # (CPUIOWR & HITMn & IHA2);

```

```

end PLD19;

```

module PLD20

title 'DRAM CONTROLLER - PLD 20
INTEL CORPORATION, August 1992'

" Generates Bank Select during EISA Burst Cycles"

PLD20 device 'E224C';

x, c, z = .X., .C., .Z.;

" Inputs

CLK	pin	2;	"CPU Input Clock
EMSTR16n	pin	27;	"Active for 16-bit ISA master
CMDn	pin	3;	"EISA Command Signal
STARTn	pin	4;	"EISA Cycle Start Signal
MSBURSTn	pin	5;	"Active for EISA Burst Cycles
DBCLK	pin	6;	"Delayed BCLK
DRAMCSn	pin	7;	"DRAM Chip Select
DRAMDISn	pin	9;	"DRAM disable signal
RESET	pin	10;	"CPU Reset Signal
HLDA	pin	11;	"CPU Hold Acknowledge Output
BNKSWTCH	pin	12;	"Bank Switch Output of EISA Cycle Tracker
WRn	pin	13;	"Write/Read Indicator
LA3	pin	16;	"Latched Address line A3
CIPn	pin	17;	"Cycle in Progress Signal
ISAMEMR	pin	26;	"ISA Read Cycle to DRAM

" Outputs

BANKSEL	pin	19;	"Bank Select for EISA Cycles
EISAMEMR	pin	20;	"EISA Read Cycle to DRAM
EISAMEMW	pin	21;	"EISA Write Cycle to DRAM
Var1	pin	23	istype 'buffer'; "State Variable
Var2	pin	24	istype 'buffer'; "State Variable
PSTROBEn	pin	25;	"Parity Strobe for EISA/ISA cycles

" state register assignments

```
sreg1 = [Var1, Var2];
S0 = [0, 0];
S1 = [0, 1];
S2 = [1, 1];
```

state_diagram sreg1

state S0 :

```
if RESET then S0 with BANKSEL := 1; endwith else
if !CIPn & !LA3 & HLDA then S1 with BANKSEL := 0; endwith else
if !CIPn & LA3 & HLDA then S2 with BANKSEL := 1; endwith
else S0 with BANKSEL := 1; endwith;
```

state S1 :

```
if RESET # CMDn # (MSBURSTn & CIPn & STARTn) then S0
with BANKSEL := 1; endwith else
if BNKSWTCH then S2 with BANKSEL := 1; endwith
else S1 with BANKSEL := 0; endwith;
```

241562-73

state S2 :

```

if RESET # CMDn # (MSBURSTn & CIPn & STARTn) then S0
  with BANKSEL := 1; endwith else
if BNKSWTCH then S1 with BANKSEL := 0; endwith
else S2 with BANKSEL := 1; endwith;

```

equations

[Var1, BANKSEL, Var2].clk = CLK;

BANKSEL.OE = HLDA;

EISAMEMR = !DRAMCSn & DRAMDISn & !CMDn & !WRn & HLDA & EMSTR16n;

EISAMEMW = !DRAMCSn & !CMDn & WRn & HLDA & EMSTR16n;

IPSTROBEn = ISAMEMR # EISAMEMR;

end PLD20;

241562-74

module PLD21

title 'DRAM CONTROLLER - PLD 21
INTEL CORPORATION, August 1992'

" Generates EADS#, FLUSH# and HOLD to CPU"

PLD21 device 'E0320';

x, c, z = .X., .C., .Z.;

" Inputs

CLK	pin	1;	"CPU input clock"
SMCLK	pin	2;	"Delayed 33-MHz CLK (skewed by 5 ns)"
EADSRIn	pin	3;	"EADS# due to ISA Read Cycles
EADSREn	pin	4;	"EADS# due to EISA Read Cycles
EBCEADSn	pin	5;	"EADS# signal generated by EBC
RESET	pin	6;	"CPU Reset Signal
SLOWHn	pin	7;	"Slow Hold Function (from Timer in ISP)
HHOLD	pin	8;	"CPU Hold Request from EBC
FLUSH	pin	9;	"Cache Flush Signal
HLDA	pin	11;	"CPU Hold Acknowledge Output

" Outputs

P5EADSn	pin	18;	"EADS# input to CPU"
P5HOLD	pin	17;	"HOLD input to CPU"
P5FLUSHn	pin	16;	"FLUSH# input to CPU"
P5INV	pin	19;	"INV input to CPU"
SHOLD	pin	15;	"
SFLUSHn	pin	14;	"
Var1	pin	13;	"State Variable"
Var2	pin	12;	"State Variable"

" State Register Assignments

sreg	=	[Var1, Var2];
S0	=	[0, 0];
S1	=	[0, 1];
S2	=	[1, 1];
S3	=	[1, 0];

EADSW = !SMCLK & HLDA & !EBCEADSn;

state_diagram sreg

```

state S0 :
    if RESET then S0 with SFLUSHn := 1; endwith else
    if FLUSH then S1 with SFLUSHn := 0; endwith
    else S0 with SFLUSHn := 1; endwith;

state S1 :
    if RESET then S0 with SFLUSHn := 1; endwith
    else S2 with SFLUSHn := 0; endwith;

state S2 :
    goto S0 with SFLUSHn := 1; endwith;

```

241562-75

equations

```

[Var1, SFLUSHn, Var2, SHOLD, P5HOLD].clk = CLK;

IP5EADSn = !EADSRIn # !EADSREn # EADSW;

SHOLD := !SLOWHn & !RESET;

P5HOLD := HHOLD # SHOLD;

IP5FLUSHn = !SFLUSHn # SHOLD;

P5INV = EADSW;

```

```

end PLD21;

```

module PLD22

title 'DRAM CONTROLLER - PLD 22
INTEL CORPORATION, August 1992'

" Generate ADS#, PCHK#, KEN#, and CPUMISS# for System (EBC)"

PLD22 device 'E0320';

x, c, z = .X., .C., .Z.;

" Inputs

CLK	pin	1;	"CPU input clock"
SMCLK	pin	2;	"Delayed 33-MHz CLK (skewed by 5 ns)"
BREQ	pin	3;	"CPU Bus Request"
HLDA	pin	4;	"Hold Acknowledge from CPU"
KENn	pin	5;	"KEN# output from Decode logic"
WRn	pin	6;	"Write/Read Indicator"
MIOn	pin	7;	"Memory/IO indicator"
ADSn	pin	8;	"ADS# from CPU"
PCHKn	pin	9;	"PCHK# from CPU"
RESET	pin	11;	"CPU Reset Signal"

" Outputs

HADSn	pin	12;	"ADS# regenerated for 33-MHz bus"
HPCHKn	pin	13;	"PCHK# regenerated for 33-MHz bus"
CPUMISSn	pin	14;	"Generated for ISP in order to include CPU in arbitration"
EBCKENn	pin	15;	"KEN# signal generated for EBC"
Var1	pin	16;	"State Variable"
Var2	pin	17;	"State Variable"
Var3	pin	18;	"State Variable"
Var4	pin	19;	"State Variable"

" State Register Assignments

```
sreg1 = [Var1, Var2];
S0 = [0, 0];
S1 = [0, 1];
S2 = [1, 1];
S3 = [1, 0];

sreg2 = [Var3, Var4];
S4 = [0, 0];
S5 = [0, 1];
S6 = [1, 1];
S7 = [1, 0];
```

state_diagram sreg1

```
state S0 :
    if RESET then S0 with HADSn := 1; endwith else
    if !ADSn & SMCLK then S1 with HADSn := 0; endwith else
    if !ADSn & !SMCLK then S2 with HADSn := 0; endwith
    else S0 with HADSn := 1; endwith;
```

```
state S1 :
```

```

    if RESET then S0 with HADSn := 1; endwith
    else S3 with HADSn := 0; endwith;

state S2 :
    if RESET then S0 with HADSn := 1; endwith
    else S1 with HADSn := 0; endwith;

state S3 :
    goto S0 with HADSn := 1; endwith;

state_diagram sreg2

state S4 :
    if RESET then S4 with HPCHKn := 1; endwith else
    if !PCHKn & SMCLK then S5 with HPCHKn := 0; endwith else
    if IPCHKn & !SMCLK then S6 with HPCHKn := 0; endwith
    else S4 with HPCHKn := 1; endwith;

state S5 :
    if RESET then S4 with HPCHKn := 1; endwith
    else S7 with HPCHKn := 0; endwith;

state S6 :
    if RESET then S4 with HPCHKn := 1; endwith
    else S5 with HPCHKn := 0; endwith;

state S7 :
    goto S4 with HPCHKn := 1; endwith;

equations

[Var1, HADSn, HPCHKn, Var2, Var3, Var4].clk = CLK;

!CPUMISSn = BREQ & HLDA;

!EBCKENn = !KENn & !WRn & !MION;

end PLD22;

```

241562-78

module PLD23

title 'DRAM CONTROLLER - PLD 23
INTEL CORPORATION, August 1992'

" Combine and generates BRDY# to CPU and CPURDY# for System (EBC)"

PLD23 device 'E0320';

x, c, z = .X., .C., .Z.;

" Inputs

CLK	pin	1;	"CPU input clock"
SMCLK	pin	2;	"Delayed 33-MHz CLK (skewed by 5 ns)"
HERDYOn	pin	3;	"Host Bus Early READY Output from EBC"
LOCRDYn	pin	4;	"READY output used when programming decode SRAM"
MRBRDYn	pin	5;	"BRDY# for CPU Read Cycles"
MWBRDYn	pin	6;	"BRDY# for CPU Write Cycles"
RESET	pin	7;	"CPU Reset Signal"

" Outputs

P5BRDYn	pin	18;	"Combined BRDY# generated for CPU"
CPURDYn	pin	16;	"
Var1	pin	13;	"State Variable"
Var2	pin	12;	"State Variable"

" State Register Assignments

```
sreg = [Var1, Var2];
S0 = [0, 0];
S1 = [0, 1];
S2 = [1, 1];
S3 = [1, 0];

HRDY = ISMCLK # (HERDYOn & LOCRDYn);
```

state_diagram sreg

```
state S0 :
    if RESET then S0 with CPURDYn := 1; endwith else
    if IP5BRDYn.FB & SMCLK then S1 with CPURDYn := 0; endwith else
    if IP5BRDYn.FB & ISMCLK then S2 with CPURDYn := 0; endwith
    else S0 with CPURDYn := 1; endwith;

state S1 :
    if RESET then S0 with CPURDYn := 1; endwith
    else S3 with CPURDYn := 0; endwith;

state S2 :
    if RESET then S0 with CPURDYn := 1; endwith
    else S1 with CPURDYn := 0; endwith;

state S3 :
    goto S0 with CPURDYn := 1; endwith;
```

equations


```

[Var1, CPURDYn, Var2].clk = CLK;

P5BRDYn = MRBRDYn & MWBRDYn & HRDY;

end PLD23;

```

241562-80

module PLD24

title 'DRAM CONTROLLER - PLD 24
INTEL CORPORATION, August 1992'

- * Write Enable Generation PLD for the DRAM Array. It also generates the Row Select
- * Signal to switch the address multiplexors from row to column address

PLD24 device 'E224C';

x, c, z = .X., .C., .Z.;

* Inputs

CLK	pin	2;	"CPU input clock"
RESET	pin	27;	"CPU Reset Signal"
CIPn	pin	3;	"DRAM Cycle in Progress"
DRAMCSn	pin	4;	"DRAM Chip Select"
WRn	pin	5;	"Write/Read Indicator"
BANKSEL	pin	6;	"Bank 0 or Bank 1 Access"
WRTPROTn	pin	7;	"Indicates Write Protected Address"
W0n	pin	16;	"Write Cycle Indication for Bank 0"
W1n	pin	9;	"Write Cycle Indication for Bank 1"
RASA0n	pin	10;	"RAS# signal for lower half of 512K in Row A"
RASA1n	pin	11;	"RAS# signal for upper half of 512K in Row A"
RASB0n	pin	12;	"RAS# signal for lower half of 512K in Row B"
RASB1n	pin	13;	"RAS# signal for upper half of 512K in Row B"

* Outputs

WE0An	pin	20;	"Write Enable for Bank 0, Row A"
WE0Bn	pin	21;	"Write Enable for Bank 0, Row B"
WE1An	pin	23;	"Write Enable for Bank 1, Row A"
WE1Bn	pin	24;	"Write Enable for Bank 1, Row B"
Var1	pin	18;	"State Variable"
Var2	pin	19;	"State Variable"
ROWSEL1	pin	25;	"Row Select to switch address multiplexors"

state_diagram [Var1]

```
state [1] :
    if RESET then [1]
    else if ICIPn & !DRAMCSn & WRn & IBANKSEL & WRTPROTn then [0]
    else [1];

state [0] :
    if RESET # W0n then [1]
    else [0];
```

state_diagram [Var2]

```
state [1] :
    if RESET then [1]
    else if !CIPn & !DRAMCSn & WRn & BANKSEL & WRTPROTn then [0]
    else [1];

state [0] :
    if RESET # W1n then [1]
    else [0];
```

equations

241562-81

```

[Var1, Var2].clk = CLK;

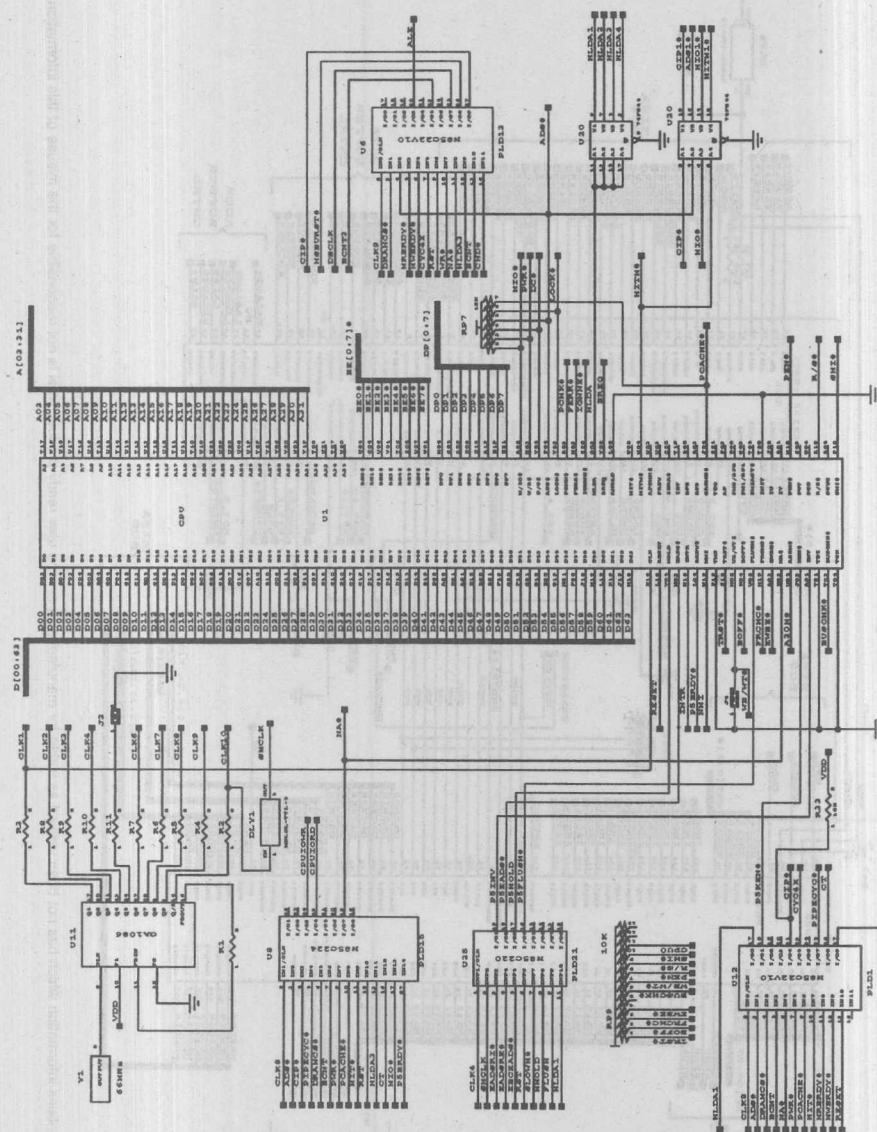
WE0An = Var1.FB;
WE0Bn = Var1.FB;
WE1An = Var2.FB;
WE1Bn = Var2.FB;

ROWSEL1 = !RASA0n # !RASA1n # !RASB0n # !RASB1n;

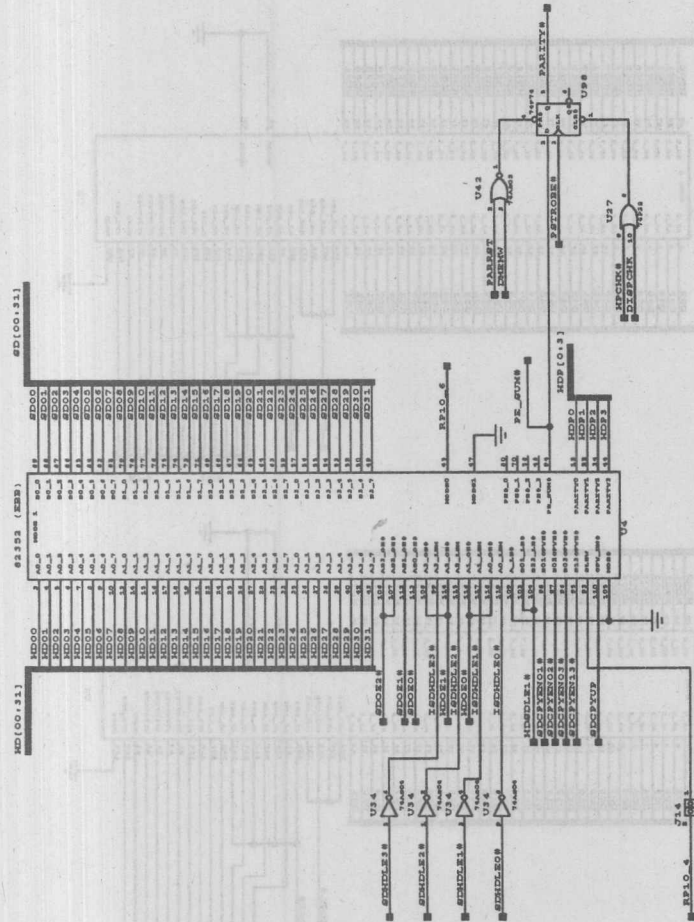
end PLD24;

```

241562-82

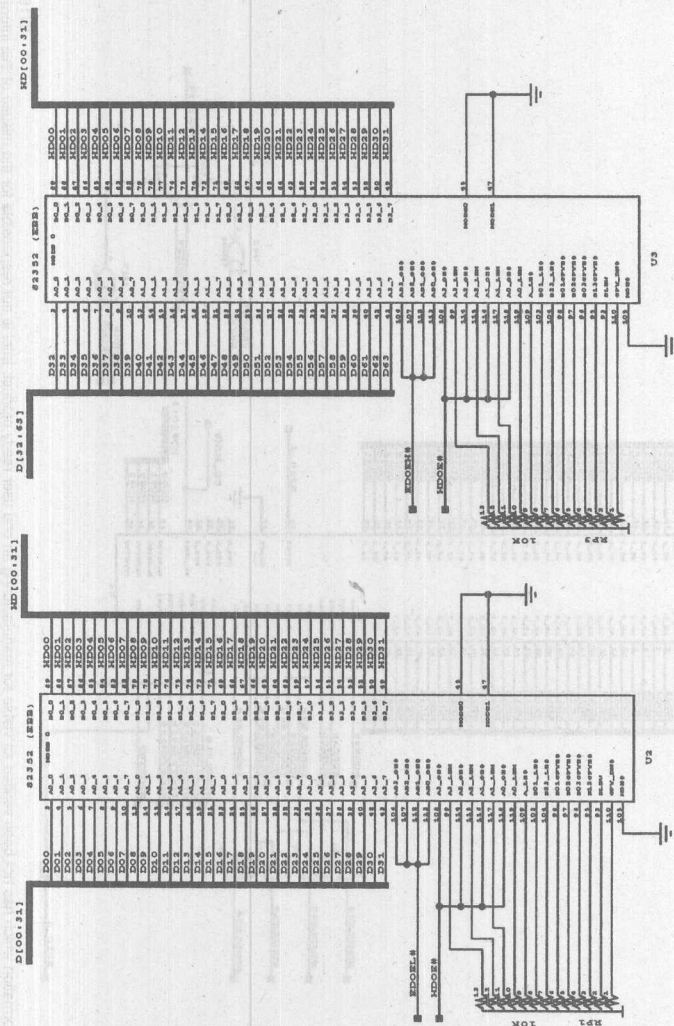


This drawing contains information which has not been verified as viable for manufacturing an end user ready product. Intel is not responsible for the misuse of this information.



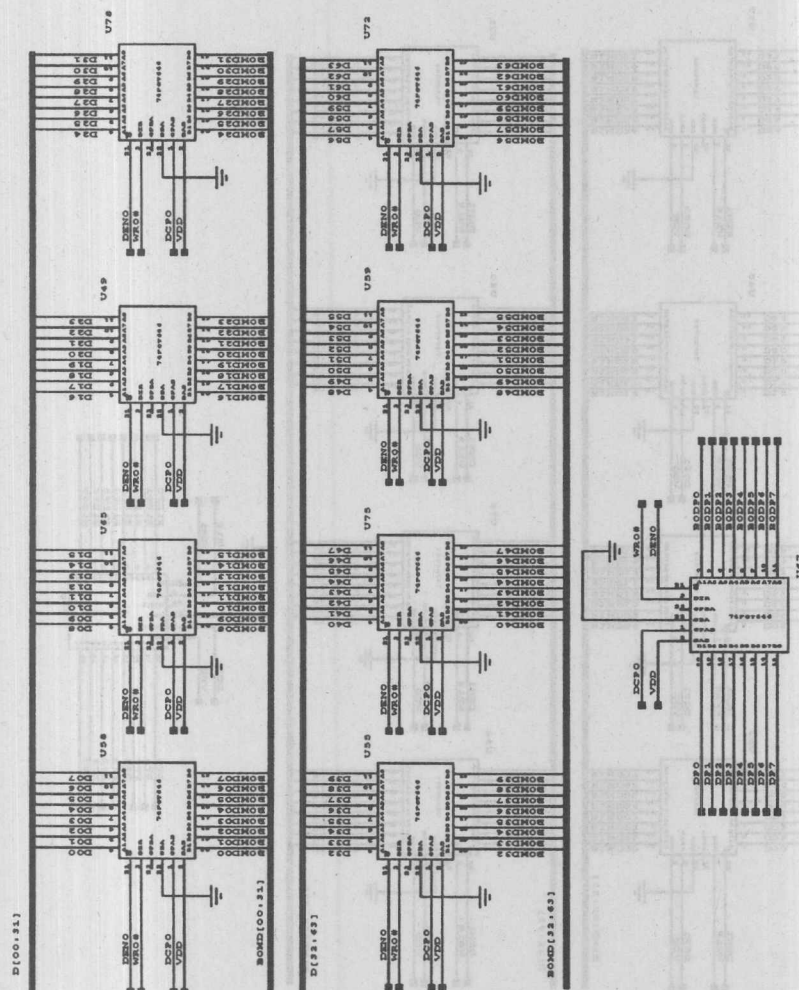
This drawing contains information which has not been verified as viable for manufacturing an end user ready product. Intel is not responsible for the misuse of this information.

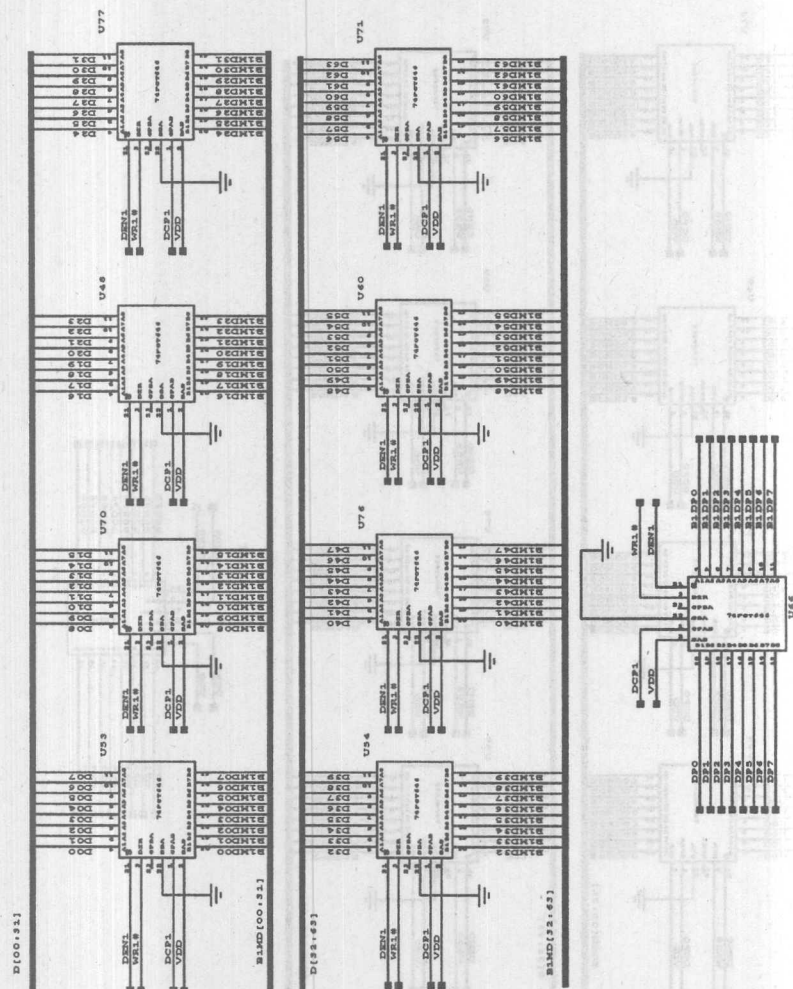
241562-85

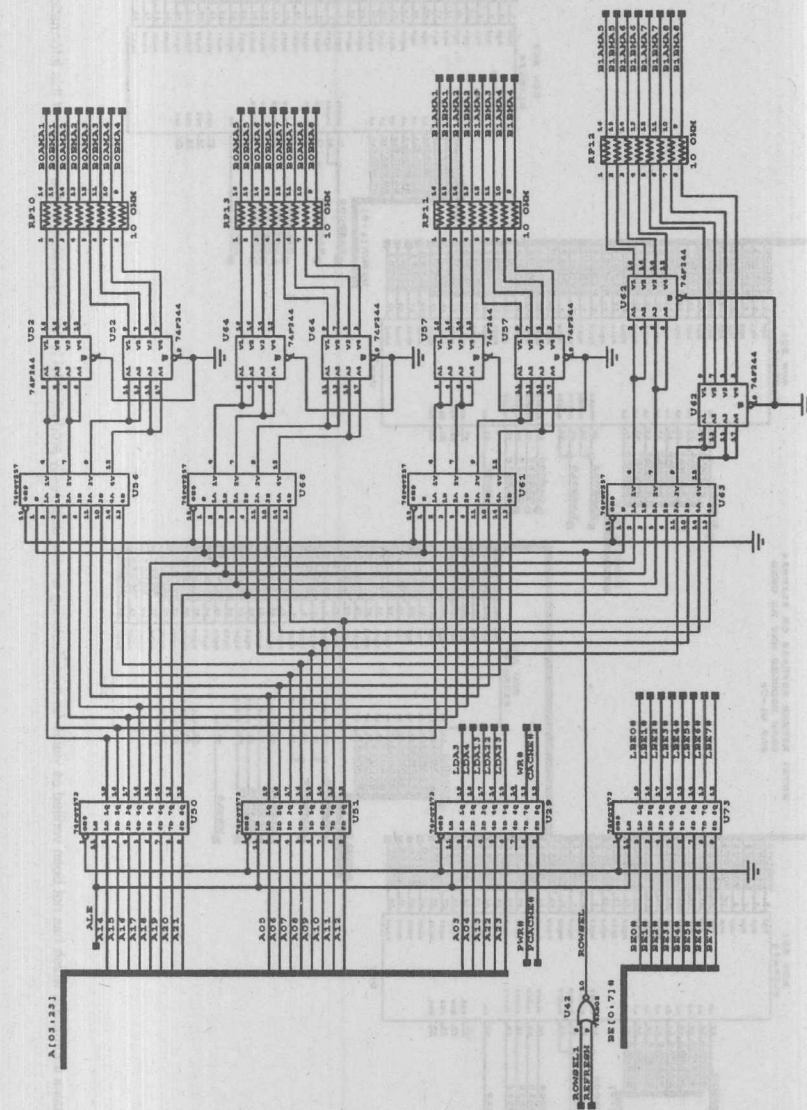


241562-86

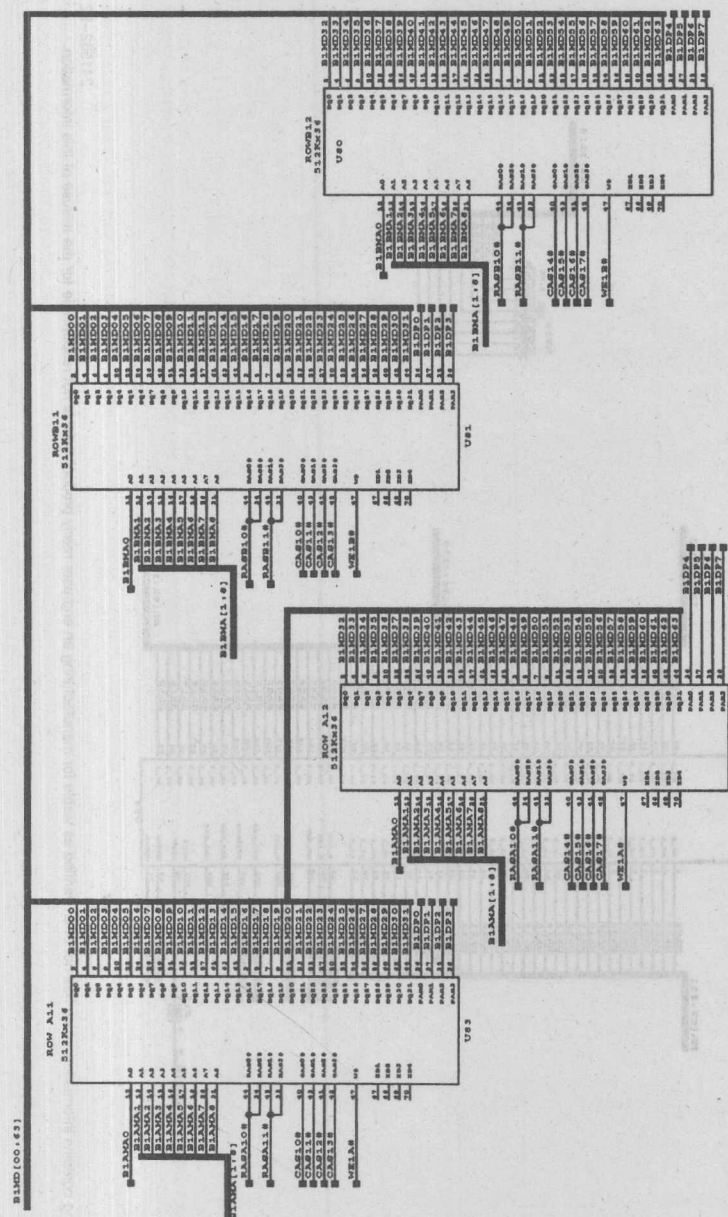
241562-87



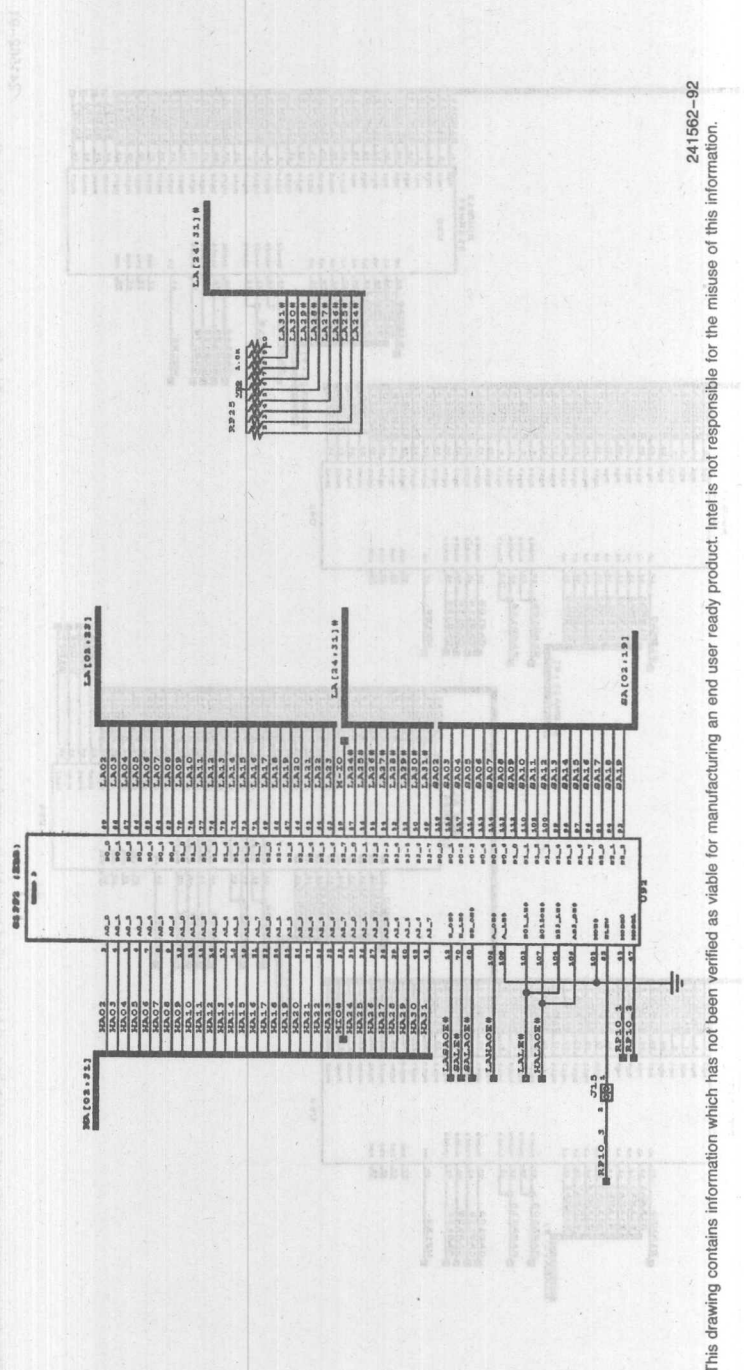


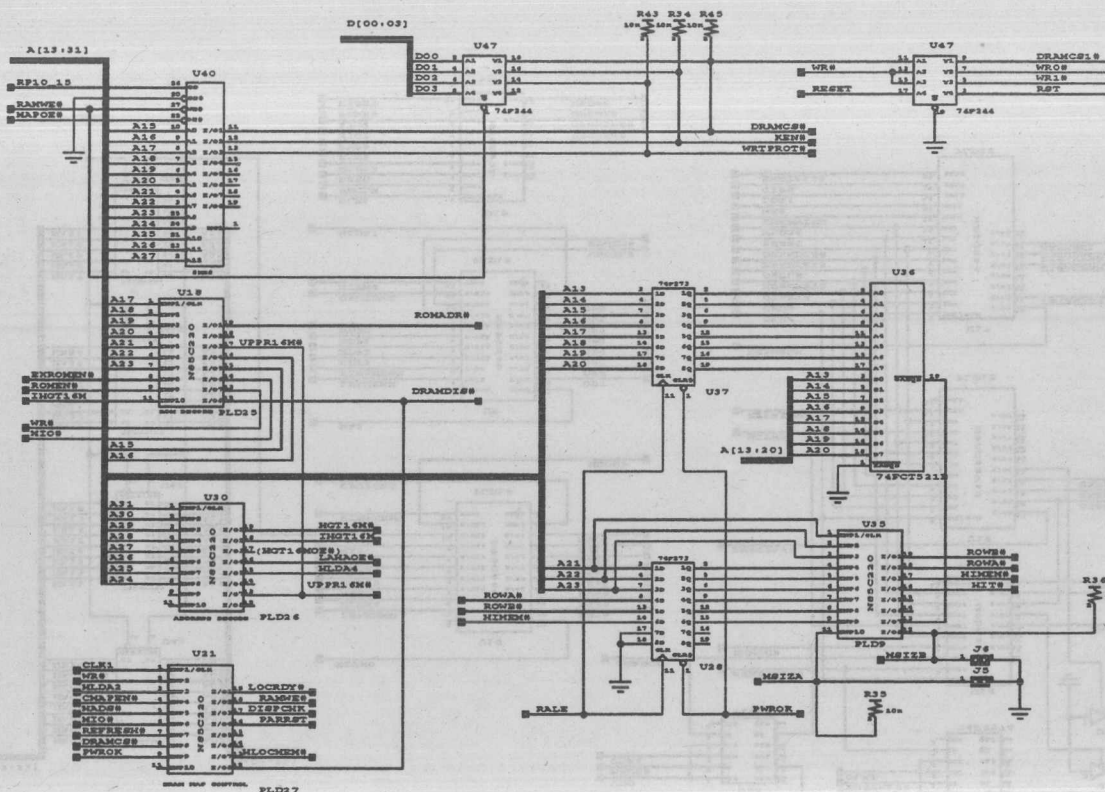


This drawing contains information which has not been verified as viable for manufacturing an end user ready product. Intel is not responsible for the misuse of this information.



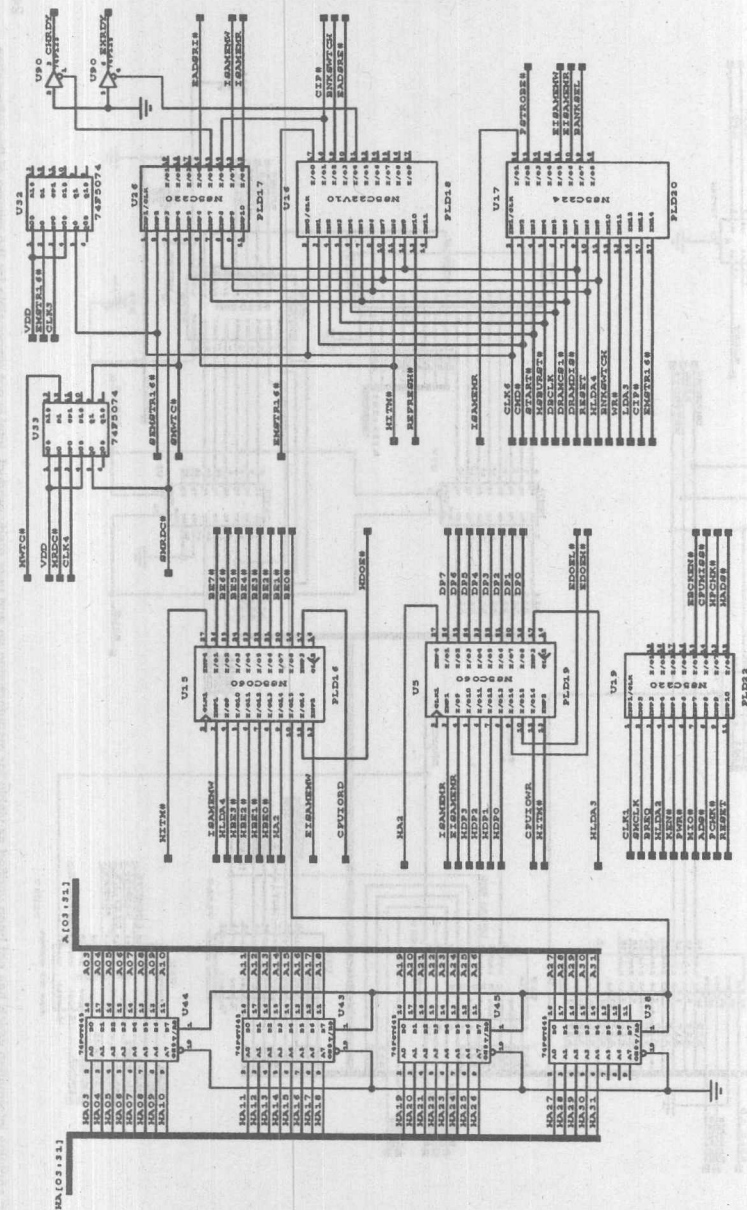
This drawing contains information which has not been verified as viable for manufacturing an end user ready product. Intel is not responsible for the misuse of this information.





This drawing contains information which has not been verified as viable for manufacturing an end user ready product. Intel is not responsible for the misuse of this information.

241562-93



241562-94

ယ





APPLICATION NOTE

3

Pentium™ Processor Clock Design

DERRICK LIN

JIM REILLY

November 1993

PRELIMINARY

Order Number: 241574-001

3-131

Pentium™ Processor Clock Design

CONTENTS	PAGE
1.0 INTRODUCTION	3-134
1.1 General Clocking Issues	3-134
2.0 Pentium™ PROCESSOR, 82496 AND 82491 SYSTEM CLOCK SPECIFICATIONS	3-135
3.0 AVAILABLE CLOCK DRIVERS	3-140
4.0 CLOCK GENERATION FOR THE Pentium™ PROCESSOR AND THE CPU-CACHE CHIP SET	3-144
4.1 Clock Generation for Fully Synchronous Systems	3-145
4.2 Clock Generation for Divided Synchronous Systems	3-145
4.3 Clock Generation for Asynchronous Systems	3-149
5.0 Pentium™ PROCESSOR WITH 256K 82496/82491 SECOND LEVEL CACHE CLOCK DISTRIBUTION DESIGN EXAMPLES	3-149
5.1 Clock Routing for the 256K CPU-Cache Chip Set	3-149
5.2 Analysis of Drivers Used in Examples	3-155
6.0 Pentium™ PROCESSOR WITH 512K 82496/82491 SECOND LEVEL CACHE CLOCK DISTRIBUTION ISSUES	3-165
7.0 CLOCK DISTRIBUTION FOR THE Pentium™ PROCESSOR WITH OTHER SECOND LEVEL CACHES ..	3-165
8.0 SUMMARY	3-165
9.0 REFERENCES	3-165
APPENDIX A. CLOCK DRIVER MANUFACTURERS	3-166

CONTENTS	PAGE
FIGURES	
Figure 1 Common Termination Techniques	3-135
Figure 2 Clock Requirements for the Pentium™ Processor and CPU-Cache Chip Set	3-137
Figure 3 An Example of an Acceptable Clock Waveform (Diodes Are Absent from the Input Model)	3-138
Figure 4 An Example of an Acceptable Clock Waveform (Diodes Are Present in the Input Model) ..	3-139
Figure 5 An Example of an Unacceptable Clock Waveform (Diodes Are Absent from the Input Model)	3-140
Figure 6 A CPU Module with the Pentium™ Processor, 82496 and 82491 CPU-Cache Chip Set	3-144
Figure 7 Examples of Clock Generation	3-145
Figure 8 Clock Generation Using Clock Doubler	3-146
Figure 9 Clock Generation Using Clock Doubler	3-146
Figure 10 Clock Generation Using Clock Divider	3-147
Figure 11 Clock Generation Using Two PLLs	3-147
Figure 12 Clock Generation Using Two PLLs	3-148
Figure 13 Pentium™ Processor, 82496 and 82491 Clock Input Models	3-150
Figure 14 CLK0 Layout for 256K Chip Set with Parity	3-151
Figure 15 CLK1 Layout for 256K Chip Set with Parity	3-152
Figure 16 CLK2 Layout for 256K Chip Set with Parity	3-153

Pentium™ Processor Clock Design

CONTENTS	PAGE
FIGURES	
Figure 17 CLK3 Layout for 256K Chip Set with Parity	3-154
Figure 18 Motorola Waveform	3-158
Figure 19 National Waveform	3-159
Figure 20 Vitesse (Slow) Waveform	3-160
Figure 21 Vitesse (Slow) Waveform (Continued)	3-161
Figure 22 Vitesse (Fast) Waveform	3-162
Figure 23 Triquint Waveform	3-163
Figure 24 Triquint Waveform (Contd.) ..	3-164

CONTENTS	PAGE
TABLES	
Table 1 Clock Signal Quality Specifications	3-136
Table 2 Clock Signal Quality Guidelines	3-136
Table 3 Clock Driver Options	3-141
Table 4 List of Clock Doubler Parts	3-148
Table 5 List of Clock Divider Parts	3-148
Table 6 Interconnect Characteristics ...	3-155
Table 7 Compilation of Simulation Data	3-156
Table 8 Series Termination Resistor Values for Each Line	3-157

1.0 INTRODUCTION

Today's high speed microprocessors place a heavy demand on clock generation and distribution. To maintain a synchronous system, well-controlled and precise clocking solutions are required. Pentium™ processor, with operating frequencies of 60 MHz and 66 MHz, has tight system clock specifications. In order to bring clock signals of acceptable quality and minimal skew to the Pentium processor and the rest of the system, system designers have to contend with high speed issues for clock distribution and limited number of precise clock driver devices. In this application note, the key issues in the design of a 60 MHz or 66 MHz clock for a Pentium processor-based system will be discussed, available clock drivers will be listed and discussed, and detailed design examples of a clock solution for the Pentium processor with 256K second-level cache subsystem, using the 82496 Cache Controller and the 82491 Cache SRAMs, are provided.

The Pentium processor, 82496 Cache Controller, and 82491 Cache SRAM form a CPU-Cache core or chip set. Along with a memory bus controller (MBC), the chip set provides a CPU-like interface for many types of memory buses.

This application note is intended for system designers concerned with clock generation and distribution for the Pentium processor and CPU-Cache chip set based systems. It reflects data collected from several quarters of characterization of the Pentium processor and experience with some of the clock driver devices, as well. This application note gives readers a good understanding of the issues and solutions of high speed clocking, particularly that for the Pentium processor. The reader should be familiar with the Pentium processor and CPU-Cache chip set electrical and mechanical specifications, *Clock Design in 50 MHz Intel486™ Systems*, and transmission line theory. If not, please read materials listed in Section 9.0 before proceeding.

1.1 General Clocking Issues

There are two major problems with distributing clock signals at 66 MHz: clock signal quality and clock skew. At high speed, one set of effects which has been minor in slower designs is now significant—the effects of transmission line. At high frequencies and fast edge rates, long traces behave like transmission lines. The “lumped” circuit assumption which assumes instantaneous signal transmission is no longer valid. Instead, signals travel in a finite time. When a transmission line is not properly terminated, one can observe severe overshoot, undershoot and ringback, all of which degrade logical signals. Bad signal quality can cause false switching or multiple switching, and can in extreme cases damage the devices. To maintain a clean clock signal, designers must consider clock driver characteristics, signal routing, load characteristics, and transmission line termination.

There are four basic ways to terminate a transmission line, series, parallel, Thevenin, and AC terminations (Figure 1). Series termination is recommended when driver output impedance is less than the transmission line characteristic impedance (true for most TTL drivers) and the line is driving a small number of devices. Series termination consumes low power and uses only one device; however, the termination method increases signal rise and fall times. Series termination ensures good signal quality by eliminating secondary reflection off the driver end. The rest of the termination methods eliminate reflection at the load end. All of the termination methods can provide good, clean clock signals at the load. Both parallel and Thevenin terminations consume a large amount of power. Thevenin termination consumes less power than parallel but requires one more device. AC termination consumes low power but adds capacitive load to the driver and delay due to RC time constant. Design examples provided with this application note use series termination. For more information on transmission line effects and design issues, please refer to [ref. 3, ref. 4, ref. 5]

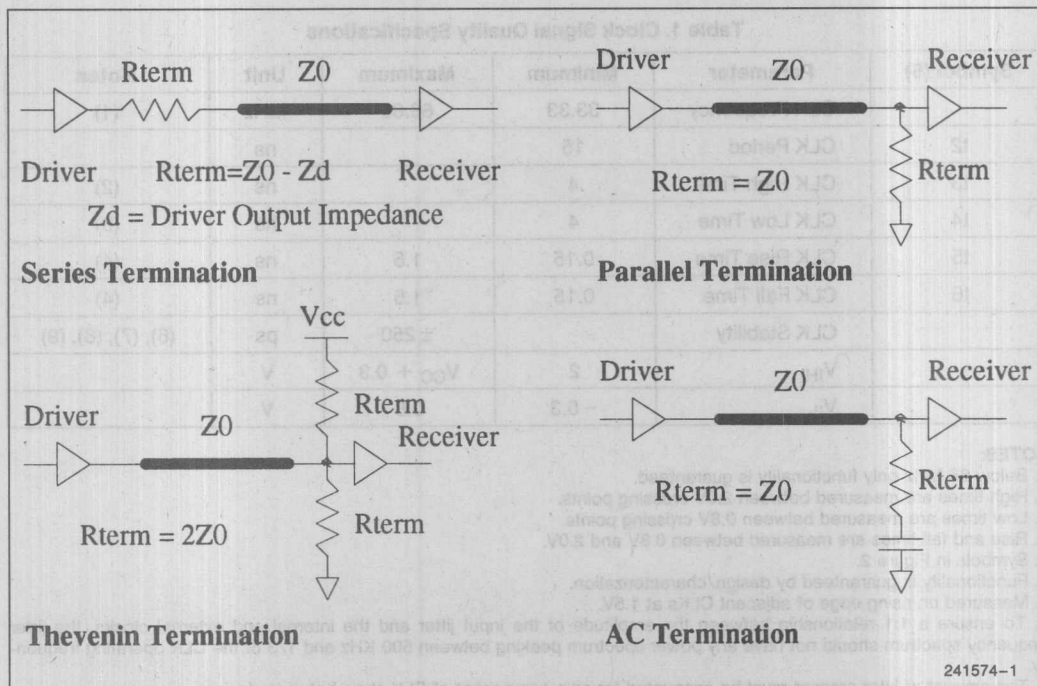


Figure 1. Common Termination Techniques

Skew is defined as the time difference between when the clock signal reaches each component. As frequency increases, there is less and less time for computation in a given clock period for a synchronous design. For a typical design, the time from one rising edge to the next is composed of the largest path-delay, setup time, propagational delay through logic elements, and skew. Clock skew then, takes away from the time available for propagational delay, thereby restricting the amount of logic done in a clock cycle. For high speed designs, skew must be minimized.

To minimize skew, designers must tune clock traces so that the propagational delay from driver through each trace to load is the same for each load. For balanced loads, tuned traces have same lengths. For unbalanced loads, trace lengths can be adjusted to make up for loading differences. If possible, designers should try to keep the loading on each clock line the same.

2.0 Pentium™ PROCESSOR, 82496 AND 82491 SYSTEM CLOCK SPECIFICATIONS

System clock specifications can be divided into 2 categories: signal quality requirements and skew specifications. Clock signal quality requirements are the same for the Pentium processor and CPU-Cache chip set. Skew specifications are only required for CPU-Cache chip set.

Signal quality requirements define boundaries for acceptable signal shapes and levels. There are two parts to signal quality requirements: signal quality specifications (Table 1) and guidelines (Table 2). Please refer to the latest revision of the Pentium processor and CPU-Cache chip set specifications for more details and for the most up-to-date information.

Table 1. Clock Signal Quality Specifications

Symbol (5)	Parameter	Minimum	Maximum	Unit	Notes
	CLK Frequency	33.33	66.66	MHz	(1)
t2	CLK Period	15		ns	
t3	CLK High Time	4		ns	(2)
t4	CLK Low Time	4		ns	(3)
t5	CLK Rise Time	0.15	1.5	ns	(4)
t6	CLK Fall Time	0.15	1.5	ns	(4)
	CLK Stability		± 250	ps	(6), (7), (8), (9)
	V _{IH}	2	V _{CC} + 0.3	V	
	V _{IL}	-0.3	0.8	V	

NOTES:

- Below 66 MHz only functionality is guaranteed.
- High times are measured between 2.0V crossing points.
- Low times are measured between 0.8V crossing points.
- Rise and fall times are measured between 0.8V and 2.0V.
- Symbols in Figure 2.
- Functionality is guaranteed by design/characterization.
- Measured on rising edge of adjacent CLKs at 1.5V.
- To ensure a 1:1 relationship between the amplitude of the input jitter and the internal and external clocks, the jitter frequency spectrum should not have any power spectrum peaking between 500 KHz and 1/3 of the CLK operating frequency.
- The amount of jitter present must be accounted for as a component of CLK skew between devices.

Table 2. Clock Signal Quality Guidelines

Parameter	Maximum	Unit	Notes
Overshoot	1.6	V	(1)
Undershoot	1.6	V	(1)
Ringback	0.8	V	(2)

NOTES:

- Overshoot (undershoot) is the absolute value of the maximum voltage above V_{CC} (or below V_{SS}). The guideline assumes the absence of diodes on the input.
- Ringback is the absolute value of the maximum voltage at the receiving pin below V_{CC} (or above V_{SS}) relative to V_{CC} (or V_{SS}) level after the signal has reached its maximum voltage level. The input diodes are assumed present.

The overshoot guideline should be used in simulations, without diodes present, to ensure overshoot (undershoot) is within the acceptable range. The ringback guideline is provided for verification in an actual system.

System designers do not have to worry about ringback if the signal does not overshoot or undershoot, respectively. Figure 2 summarizes clock waveform requirements listed in Table 1.

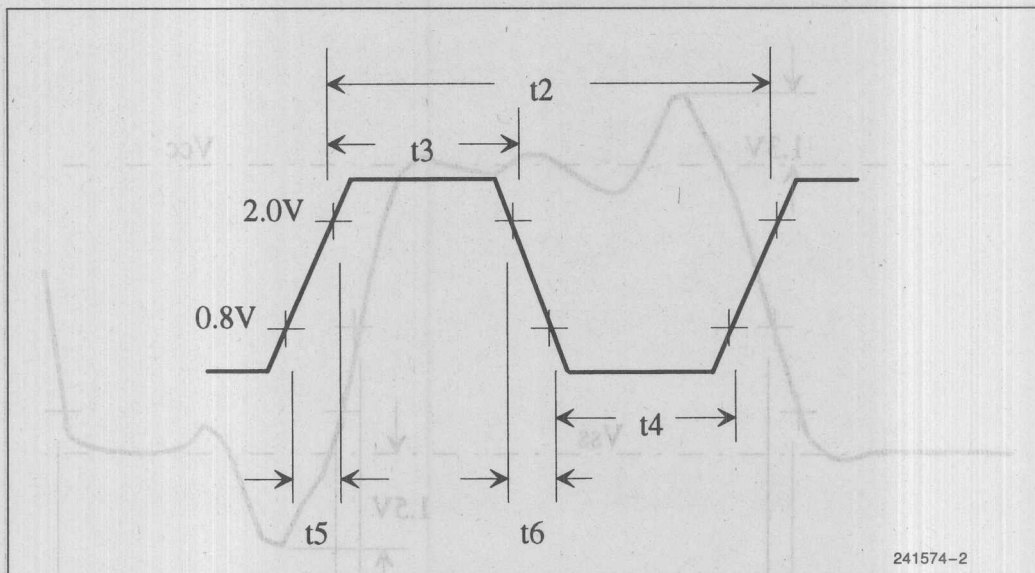


Figure 2. Clock Requirements for the Pentium™ Processor and CPU-Cache Chip Set

Figure 3 to Figure 5 illustrates examples of acceptable and unacceptable clock waveforms. Waveform in Figure 3 is for an input model without diodes. Waveform in Figure 4 is for an input model with diodes. The diodes clamp the voltage and prevent it from going more than a diode drop above V_{CC} or below V_{SS} . Waveform

in Figure 5 is for an input model without diodes. The waveform is not acceptable for several reasons. It violates the minimum low time specification (4 ns), the maximum fall time specification (1.5 ns), and it does not follow the maximum undershoot guideline (1.6V).

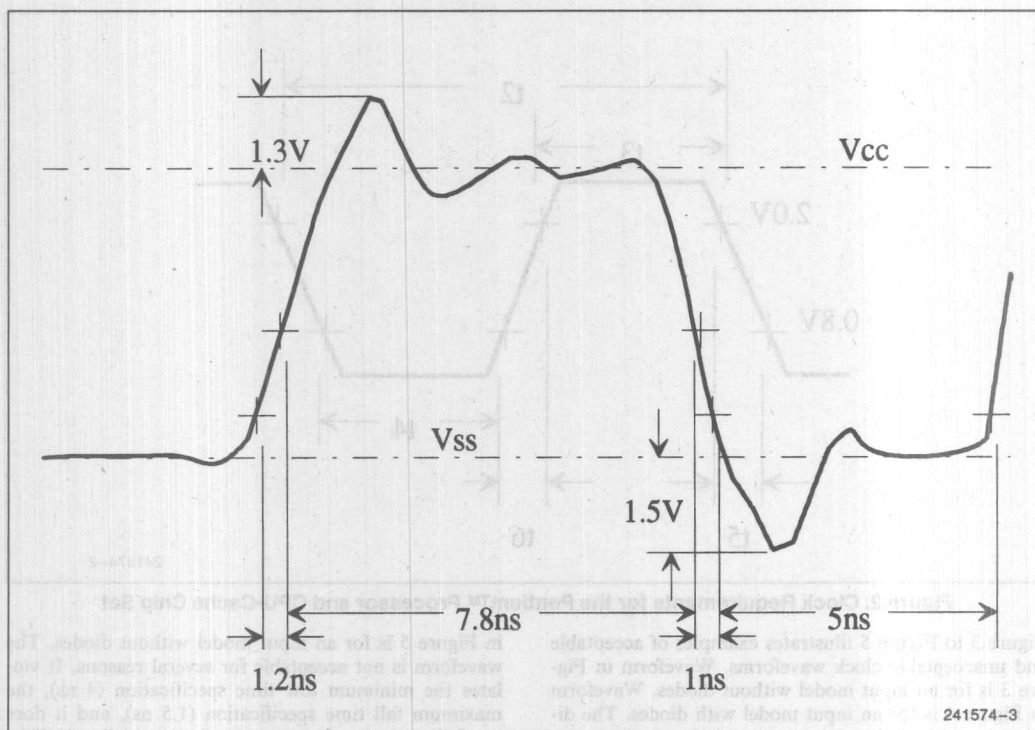


Figure 3. An Example of an Acceptable Clock Waveform (Diodes are Absent from the Input Model)

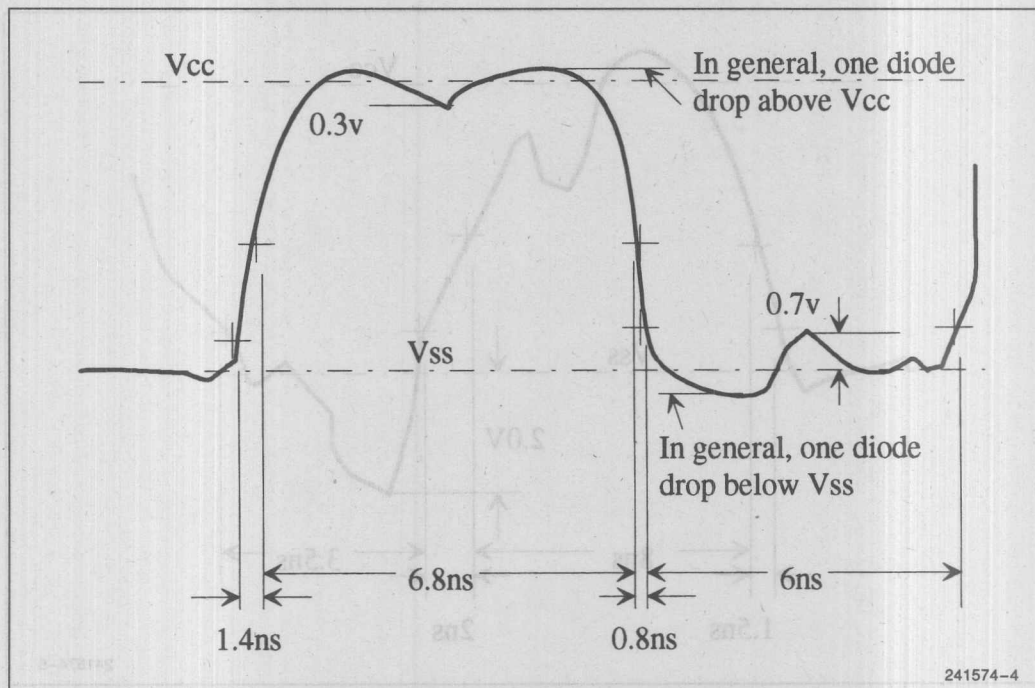


Figure 4. An Example of an Acceptable Clock Waveform (Diodes are Present in the Input Model)

above 66 MHz. Preliminary data sheets show that some items listed in Table 3 meet the CPU or CPU-Cache chip set requirements. The specifications listed are based on preliminary data provided by each company and may be subject to change. Intel's should contact each company for the latest specifications and availability. Intel's qualification has been done by simulating an example clock system using output models supplied by a subset of the companies listed, along with internal models and preliminary clock input models of the CPU-Cache chip set. For more details on the simulation and example timing, please see Section 5.0. Intel has been and will be working closely with the listed companies to ensure they have the latest specifications for the Pentium processor. With published preliminary data shown, all the listed parts meet either the CPU or the chip set clock specifications (including the signal quality and slew specifications). These critical indicators mean incentives for data sheets and sample availability.

Clock signals for the CPU-Cache chip set are measured at 0.9V (3V) and 1.0V on the rising edge. Worst case skew between the Pentium processor and the 82495 is 0.2 ns and worst case skew between any 82495 and other the Pentium processor or the 82495 is 0.7 ns.

3.0 AVAILABLE CLOCK DRIVERS

Intel has held discussions with many clock driver companies. The intent has been to enable these companies to offer clock driver solutions that meet the Pentium processor specifications. It has also been to ensure that the upper set of these companies can provide support and distribution worldwide on a schedule that closely matches the Pentium processor's availability. Based on information available, Table 3 lists a number of companies who are planning to offer solutions to meet these requirements. All the clock drivers listed in Table 3 have a maximum output frequency equal to

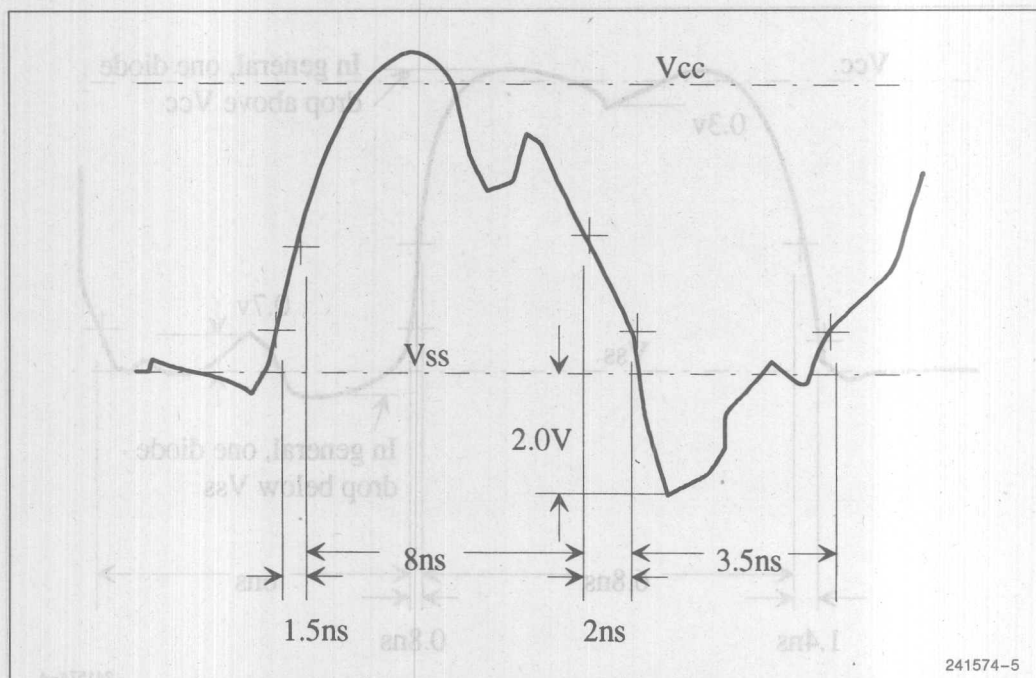


Figure 5. An Example of an Unacceptable Clock Waveform (Diodes Are Absent from the Input Model)

Clock skews for the CPU-Cache chip set are measured at 0.8V, 1.5V, and 2.0V on the rising edge. Worst case skew between the Pentium processor and the 82496 is 0.2 ns, and worst case skew between any 82491 and either the Pentium processor or the 82496 is 0.7 ns.

3.0 AVAILABLE CLOCK DRIVERS

Intel has held discussions with many clock driver component companies. The intent has been to enable these companies to offer clock driver solutions that meet the Pentium processor specifications. It has also been to ensure that the super set of these companies can provide support and distribution worldwide on a schedule that closely matches the Pentium processor's availability. Based on information available, Table 3 lists a number of companies who are planning to offer solutions to meet these requirements. All the clock drivers listed in Table 3 have maximum output frequency equal or

above 66 MHz. Preliminary data sheets show that solutions listed in Table 3 meet the CPU or CPU-Cache chip set requirements. The specifications listed are based on preliminary data provided by each company and may be subject to change. Designers should contact each company for the latest specifications and availability. Some evaluation has been done by simulating an example clock layout using output models supplied by a subset of the companies listed, along with interconnect models and preliminary clock input model of the CPU-Cache chip set. For more detail on the simulations and example routing, please see Section 5.0. Intel has been and will be working closely with the listed companies to ensure they have the latest specifications for the Pentium processor. With published preliminary data sheets, all the listed parts meet either the CPU or the chip set clock specifications (including the signal quality and skew specifications). Please contact individual manufacturers for data sheets and sample availability.

Table 3. Clock Driver Options

Mfgr	Part #	Driver Type	Level In/Out	Pin-to-Pin Skew (ns)	Part-to-Part Skew (ns)	t_r/t_f (0.8V–2.0V) (ns)	Clock Stability	# of Outputs (per pkg)	Spec'd Loading	Max. Freq.
Intel Spec			TTL inputs	(1)		1.5/1.5	± 250 ps (2)			66 MHz and 60 MHz
AMCC	SC35XX-1	Buffer	PECL or TTL/TTL	0.5	1.0	1.5/1.5		20 Outputs Which Vary with Part #	10 pF	80 MHz
	SC44XX-80	PLL	TTL/TTL	± 0.2	1.0 (9)	1.5/1.5		6–12 Outputs	35 pF	80 MHz
AT&T (7)	DA400	PLL	PECL or TTL/TTL	0.2	0.5	1.5/1.5		8@1, 0.5X (Prog Shift)	50 pF	100 MHz
Cypress	CY7B991	PLL	TTL/TTL	0.5	1.2 (6)	1.5/1.5	0.5%	4@1X 4@1, 0.5, 0.25X	50Ω/ 30 pF	80 MHz
ICS	ICS2686	PLL		0.5	0.6	1.5/1.5		5@1, 0.5X	20 pF	
Intel	85C224-100	Divider/Buffer/PLD	TTL/CMOS	Divider 0.4 Buffer 0.5	NA	Divider 1.2/1.1 Buffer 1.4/1.1		8 @ +1X, –1X, 0.5X	70Ω/ 50 pF	Divider 100/50 Buffer 133
Intel	85C224-10	Buffer/Divider/PLD	TTL/CMOS	Divider 0.4 Buffer 0.5	NA	Divider 12./1.1 Buffer 1.4/1.1		8 @ +1X, –1X, 0.5X	70Ω/ 50 pF	Divider 58/29 Buffer 100
Motorola	MC10H646	Buffer	PECL or TTL/TTL	0.5	1.0	1.2/1.2	NA	8	50Ω/ 50 pF	100 MHz
	88915	PLL	TTL/CMOS	0.5	NA	2.5/2.5(11)	NA	5@1X 1@2X 1@.5X 1 Inverted X	50Ω	66 MHz

Table 3. Clock Driver Options (Continued)

Mfgr	Part #	Driver Type	Level In/Out	Pin-to-Pin Skew (ns)	Part-to-Part Skew (ns)	t_r/t_f (0.8V-2.0V) (ns)	Clock Stability	# of Outputs (per pkg)	Spec'd Loading	Max. Freq.
National	CGS74CT2524	Buffer	TTL/CMOS	0.45	NA	1.5/1.5		4	50 pF	100 MHz
	CGS74CT2527	Buffer	TTL/CMOS	0.45	NA	1.5/1.5		8	50 pF	100 MHz
	CGS74B2528	Buffer	TTL/TTL	0.55	NA	1.5/1.5		10	50 pF	70 MHz
Pioneer	PI6B2407	PLL	TTL/TTL	±0.25	NA	1.5/1.5	100 ps	12@1, 0.5, 2X (Prog Shift)		80 MHz
TI (8)	CDC328	Buffer	TTL/TTL	0.7	NA	1.2/0.5	NA	6	500Ω/ 50 pF	Not Spec'd.
Triquint (10)	GA1085	PLL	TTL/TTL	0.25	NA	1.4/1.4	75 ps (typ.)	5@1X 4@0.5X 2@0.5X (Prog Shift)	50Ω	66 MHz
	GA1086	PLL	TTL/TTL	0.25	NA	1.4/1.4	75 ps (typ.)	9@1X 1@0.5X	50Ω	66 MHz
	GA1087	PLL	TTL/TTL	0.25	NA	1.4/1.4	75 ps (typ.)	6@1X 4@0.5X	50Ω	66 MHz
Vitesse	VSL4485	PLL	TTL/TTL	0.5	NA	1.5/1.5		6@1X 2@1, 2, 4X	50 pF	70 MHz
	VSL4586	PLL	TTL/TTL	0.5	NA	1.5/1.5		2@1X 6@1, 2, 4X	50 pF	70 MHz

NOTES:

- 0.7 ns between Pentium processor-82491, 82496-82491, 82491-82491. 0.2 ns between Pentium processor-82496. Assumed 0.5 ns between clock driver outputs, leaving 0.2 ns for routing or trace skew.
- See complete specification in Table 1 or the data book.
- Manufacturers listed in alphabetical order.
- Contact manufacturers for price and availability information.
- Intel does not guarantee specifications for other manufacturer's devices. All clock driver specifications listed were provided by the manufacturer and are subject to change. Designers should contact the manufacturer for the latest specification/data sheet information.
- As low as 0.75 ns in some configurations.
- First samples in March '93. Specifications may improve during characterization.
- Other Solutions are under development. Contact TI for preliminary details.
- Maximum phase error quoted in the manufacturer's data sheet for the entire frequency range.
- Other configurations available. Contact Triquint for details.
- Between 0.2 V_{CC} and 0.8 V_{CC}. Contact Motorola for details between 0.8 and 2.0V.

AMCC offers the SC35XX-1 series of buffered clock drivers and the SC44XX-80 series of PLL based clock drivers. The SC35XX-1 series must be driven with a TTL or PECL 2X frequency input. Each member of the series provides 20 outputs. Depending on the specific part within the series, these 20 outputs can be configured to provide the primary frequency, 1/2, or 1/4 the primary frequency. The SC3502-1 even provides 5 inverted outputs of the primary frequency. The SC44XX-80 series must be driven with a TTL input. The PLL design allows for very low skew (± 200 ps) between the outputs. Different members of the series offer different numbers and configurations of outputs. Between 4 and 8 outputs are available at the primary frequency. These devices also allow a subset of the outputs to be configured for 1/2X or 2X the primary frequency. In addition, the PLL allows the outputs to be skewed in phase from one another.

AT&T DA400 is a PLL clock driver. Its inputs can be driven by TTL or PECL levels. Eight outputs are provided. They can be configured for the primary frequency or 1/2X the primary frequency. In addition each output has a programmable delay line which allows 1/32 or 1/64 increments of the clock period of delay between outputs.

Cypress's CY7B991 is a PLL clock driver. It requires a TTL input and is able to drive 8 outputs. A subset of the outputs can be configured as 1/2X, 1/4X, or inverted outputs. As with other PLL solutions, the skew between outputs is small and the outputs can be configured for a fixed amount of delay or skew between outputs.

ICS's ICS2686 is a PLL clock driver. Five outputs are available. Both primary and 1/2X frequencies are available. The ICS2686 has been designed to work with the 74ABT240 type buffer to provide more than 5 outputs. A unique feature of the ICS2686 is the multiple feedback inputs. This feature allows synchronizing multiple outputs at their destination or load with the input clock.

Intel's 85C224-100 is a "20V8" architecture programmable logic device. From its TTL inputs it provides 8 TTL outputs which can be configured to provide 1X, 1X inverted, and 1/2X versions of the primary frequency, in any combination. When programmed to function as a frequency divider, the primary frequency can be as high as 100 MHz and the 1/2X frequency outputs will

maintain output skew below 400 ps. When programmed to operate as a straight 1X buffer, it supports frequencies of up to 133 MHz with less than 500 ps of output skew. The 85C224-100 provides a combination of superior output signal quality including fast rise and fall times and low output skew. A particularly unique feature of the 85C224-100 is in its programmable logic circuitry. Its flexibility satisfies programmable logic needs such as control line signals and widespread glue logic. With this minimized output skew PLD, a single 28-pin PLCC can provide low output skew clock distribution, frequency division, and programmable logic; for the low price of a 20V8 PLD.

Motorola offers both a buffered and a PLL clock solution. Motorola's 10H646 is a buffered clock driver. It offers both TTL and ECL inputs which supports backplane routing using ECL levels. The clock driver's outputs are clamped to 3V, not V_{CC} . 10H646's output stage has similar rise and fall output resistances. Similar rise and fall output resistances makes series termination easier since the termination resistance is the difference between the characteristic impedance of the transmission line connecting the output to the load and the driver's output impedance. 10H646 has 8 1X outputs. As a straight buffer, 10H646 does not offer any multiples of the input besides 1X. The Motorola 88915 is a PLL clock driver. It provides a 0.5 ns skew between outputs. The 88915 provides 5 1X outputs along with 1 2X, 1 0.5X, and 1 inverted X outputs.

National's clock buffers are packaged to function reliably at high frequencies. Their output rise and fall resistances are approximately equal. The CGS74CT2524 and 2527 provide 0.45 ns of output skew. The CGS74CT2528's output skew, 0.55 ns, allows for only 0.15 ns skew due to board traces or any unbalanced loading effects when using the 82496/82491 cache, however this amount may be sufficient for other cache solutions. These parts offer a range of 4, 8, and 10 outputs. The CGS74CT2524 and 2527 have CMOS level outputs, which transition from rail to rail.

Pioneer's PI6B2407 is a PLL clock driver. From its TTL input, it provides twelve TTL outputs, which can be configured to operate at 1X or 2X the input frequency. In addition, the outputs can be phase adjusted from the input clock. The PI6B2407 is able to provide ± 0.25 ns of skew between outputs while maintaining the fast 1.5 ns rise and fall times.

3

TriQuint's GA1086 is a Gallium Arsenide-based product. It takes a 66 MHz input and produces nine 66 MHz outputs and one 33 MHz output. The availability of a low skew 33 MHz output facilitates clock distribution for systems that have synchronous 33 MHz memory buses. Since the part is phase-lock-loop based, one of the outputs can be fed back to the input so that all the outputs are synchronized with the input clock. Such a set up is ideal for cascading clock drivers to achieve maximum fanout. The specified output skew of the GA1086 is 0.25 ps, the smallest skew number available. Triquint also offers the GA1085 and GA1087. These products are similar to the GA1086, however, they offer different combinations of outputs between 1X and 0.5X.

66 MHz signals with low skew, for example, the clock input frequency of the VSL4485 can be 33 MHz. For the chip set application, two 66 MHz outputs are not enough, and thus cascading another driver is necessary. Alternatively, the input can be 66 MHz and all of its outputs can be at 66 MHz. It offers 0.5 ns output skew, and a low effective delay. In addition, VSL4485 can generate programmable, multiple phase relationships among its outputs.

Clock generation is the generation of copies of clock signals from a signal oscillator or any other source which then are distributed to the various loads. The function of a clock driver is to generate multiple copies of clocks from a single source. In general, Pentium processor-based systems have three types of memory interface: fully synchronous, divided synchronous, and asynchronous. Each interface requires different methods of clock generation. The basic setup of a processor card is illustrated symbolically in Figure 6. Depending on the configuration, the Clock In signal can come from the memory bus or a separate oscillator.



4.1 Clock Generation for Fully Synchronous Systems

A fully synchronous system is one which everything in the system runs synchronous to the CPU. In particular, the memory bus interface is synchronous to the CPU. In Figure 6, the memory bus is at 66 MHz, synchronous to the CPU module. Clock In signal must be synchronous to the memory bus. Clocking for this case involves the generation of tightly controlled copies of clock signals that are distributed to all the clocked parts. The task of clock generation and distribution is the most difficult for this type of set-up. All copies of clock signals must come from a single source, and must be deskewed appropriately. For Pentium processor-based systems that run at 66 MHz, the most critical parameter in choosing a clock driver is its output skew, as well as its part-to-part skew if more than one driver is needed. Since all the clock signals are at 66 MHz, only 1x outputs are needed. All of the drivers listed in Section 3.0 can be used here.

For a fully synchronous configuration, it is likely that a single clock driver cannot provide enough copies of clock signals. Then, some kind of cascading of drivers is necessary. Figure 7 shows two ways of clock generation by cascading drivers. Tskew is the total worst case skew at outputs of CD2 and CD3. Tpp23 is the worst case part-to-part skew between CD2 and CD3. Tos2 is the worst case output skew of CD2, assuming the worst case output skew of CD3 is the same as Tos2. Tos1 is the worst case output skew of CD1. Ttol2 is the feedback tolerance of CD2. Feedback tolerance is the phase tolerance between the feedback input and the reference clock. Typically, Ttol2 is a small number. For the examples in Figure 7, it is assumed that only the second level drivers feed the clock signals to the loads. Otherwise, for part a, signals from CD2 will be later than signals from CD1 by the propagational delay of CD2 which is typically between 6 ns to 8 ns.

For the examples in Figure 7 clock signals for the CPU-Cache chip set must be derived from one clock driver outputs only so that the 0.2 ns and 0.7 ns skew specifications can be met. In part a, Tskew, the sum of Tpp23, Tos2, and Tos1 is the worst case skew which is the skew between an output of CD2, and an output of CD3. The output skew of CD1 (Tos1) causes the inputs to CD2 and CD3 to arrive at different times. The difference in propagational delay which is Tpp23, further skews the outputs of CD2 and CD3. If the part-to-part skew does not include output skew, different outputs from CD2 and CD3 can also be skewed by the output skew. For part b, Tskew, the sum of Ttol2, Tos2, and Tos1, is also the worst case skew between the outputs of CD2 and the outputs of CD3. Once again, Tos1 causes

the inputs to CD2 and CD3 to arrive at different times. The feedback in CD2 synchronizes all its outputs in the input. The feedback output of CD2 is different from the input reference clock only by Ttol2. All the other outputs are further skewed from the feedback output by Tos2. The analysis for CD3 is the same.

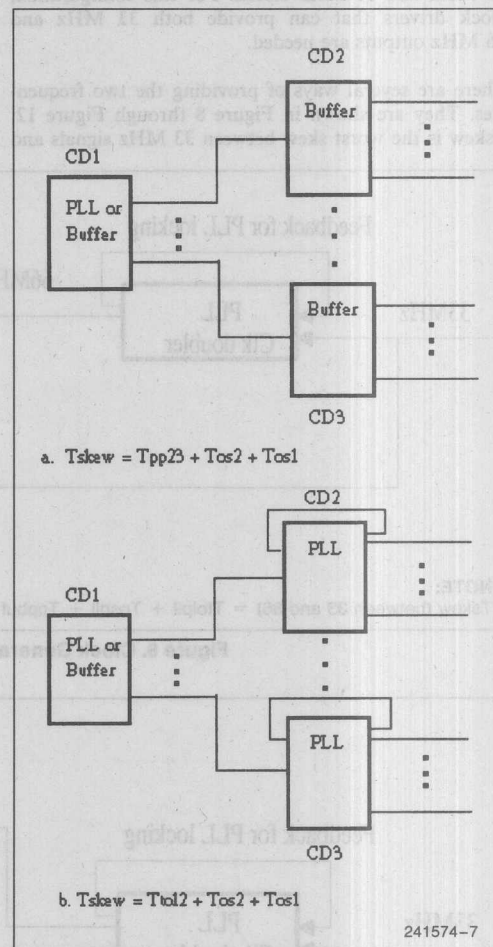


Figure 7. Examples of Clock Generation

4.2 Clock Generation for Divided Synchronous Systems

For a divided synchronous system, the memory bus is at half the speed of the CPU-Cache chip set; i.e.,

the memory bus runs at 33 MHz for the Pentium processor or the CPU-Cache chip set based systems. A 33 MHz reference clock (Clock In) can come from the backplane from which all the clocks serving the CPU-cache module (Figure 6) must be synchronized. The memory bus controller (MBC) itself requires both 33 MHz and 66 MHz clocks. For this configuration, clock drivers that can provide both 33 MHz and 66 MHz outputs are needed.

There are several ways of providing the two frequencies. They are shown in Figure 8 through Figure 12. Tskew is the worst skew between 33 MHz signals and

66 MHz signals. The skews among 66 MHz signals or among 33 MHz signals are simply the output skew of the driving devices. Ttolpll is the PLL CLK doubler or PLL CLK divider's feedback tolerance. Tosppll is the PLL CLK doubler or PLL CLK divider's worst case output skew. Tppbuf is the worst case part-to-part skew of the second level buffers. Those buffers can be phase-lock-loops also in which case Tppbuf is the feedback tolerance of the PLLs if feedback is used. Tosbuf is the worst case output skew of the second level buffers. Tos1 is the output skew of CD1, Ttol2 is the feedback tolerance of CD2, and Tos2 is the output skew of CD2.

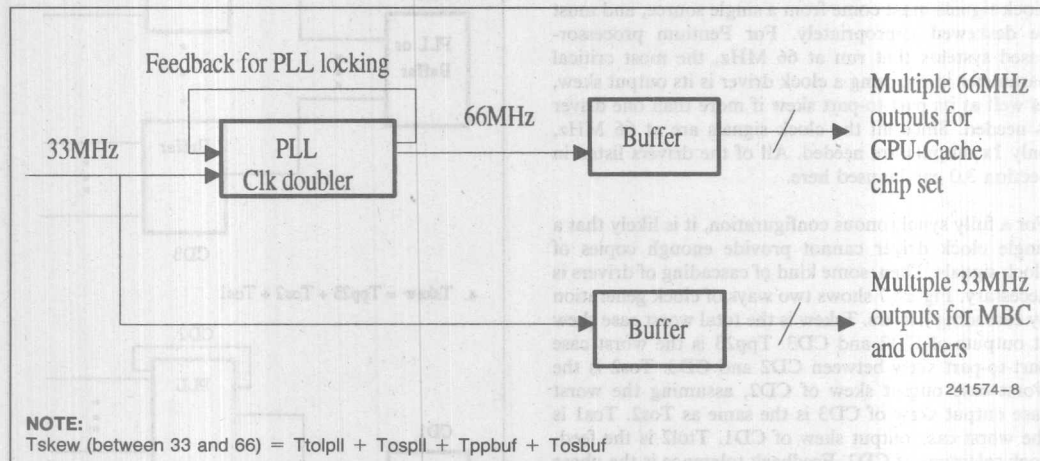


Figure 8. Clock Generation Using Clock Doubler

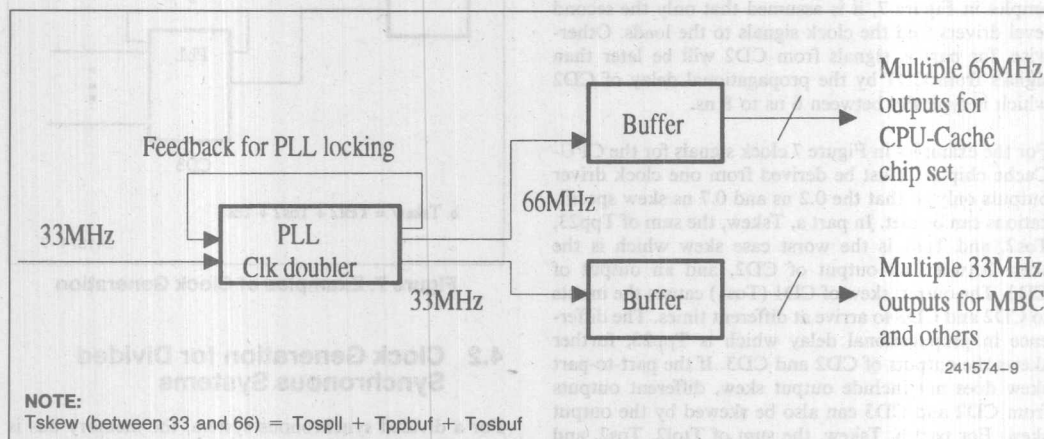


Figure 9. Clock Generation Using Clock Doubler

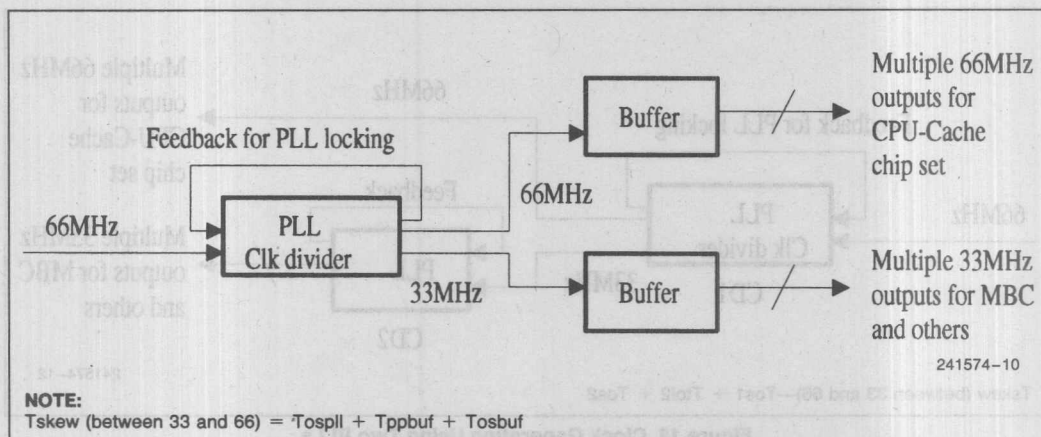


Figure 10. Clock Generation Using Clock Divider

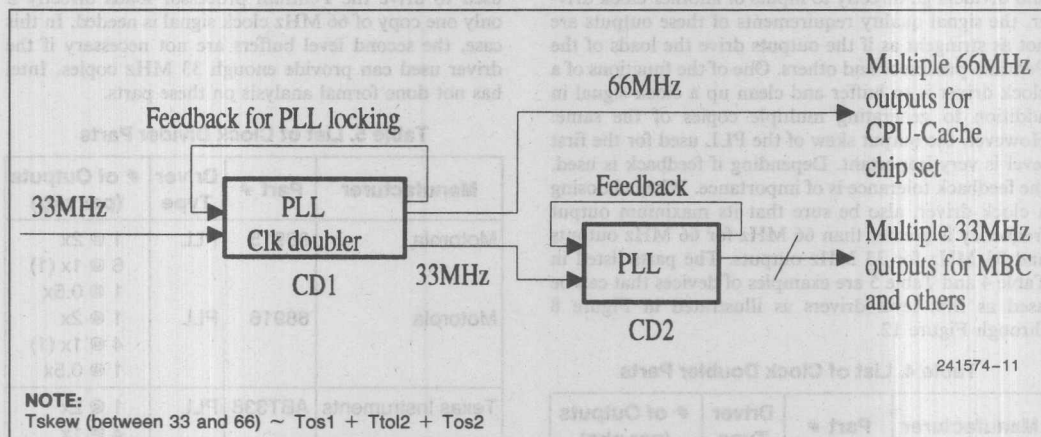


Figure 11. Clock Generation Using Two PLLs

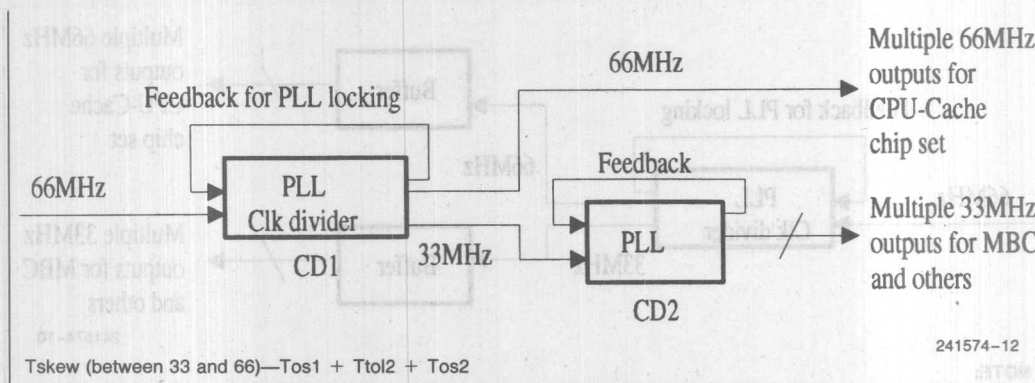


Figure 12. Clock Generation Using Two PLLs

Since the outputs of the first level PLL CLK doublers and dividers go directly to inputs of another clock driver, the signal quality requirements of these outputs are not as stringent as if the outputs drive the loads of the Pentium processor and others. One of the functions of a clock driver is to buffer and clean up a clock signal in addition to generating multiple copies of the same. However, the output skew of the PLL used for the first level is very important. Depending if feedback is used, the feedback tolerance is of importance. When choosing a clock driver, also be sure that its maximum output frequency is greater than 66 MHz for 66 MHz outputs and 33 MHz for 33 MHz outputs. The parts listed in Table 4 and Table 5 are examples of devices that can be used as first level drivers as illustrated in Figure 8 through Figure 12.

Table 4. List of Clock Doubler Parts

Manufacturer	Part #	Driver Type	# of Outputs (per pkg)
Motorola	88915	PLL	1 @ 2x 6 @ 1x (1) 1 @ 0.5x
Motorola	88916	PLL	1 @ 2x 4 @ 1x (1) 1 @ 0.5x
TI	ABT338	PLL	1 @ 2x 4 @ 1x 1 @ 0.5x
Vitesse	VSL4485	PLL	6 @ 1x 2 @ 1, 2, or 4x

NOTES:

1. One of the outputs is inverted.
2. This list is not meant to be complete. Other solutions may be available.

The phase-lock-loop drivers listed in Table 4 can be used to drive the Pentium processor loads directly if only one copy of 66 MHz clock signal is needed. In this case, the second level buffers are not necessary if the driver used can provide enough 33 MHz copies. Intel has not done formal analysis on these parts.

Table 5. List of Clock Divider Parts

Manufacturer	Part #	Driver Type	# of Outputs (per pkg)
Motorola	88915	PLL	1 @ 2x 6 @ 1x (1) 1 @ 0.5x
Motorola	88916	PLL	1 @ 2x 4 @ 1x (1) 1 @ 0.5x
Texas Instruments	ABT338	PLL	1 @ 2x 4 @ 1x 1 @ 0.5x
Texas Instruments	ABT337	Buffer	4 @ 1x 4 @ 0.5x
Texas Instruments	ABT339	Buffer	4 @ 1x 4 @ 0.5x
TriQuint	GA1086	PLL	9 @ 1x 1 @ 0.5x

NOTES:

1. One of the outputs is inverted.
2. This list is not meant to be complete. Other solutions may be available.

Table 5 lists examples of clock drivers that offer divided by 2 outputs. These devices can be used as the first level drivers illustrated in Figure 8 through Figure 12. Depending on the number of 66 MHz copies and 33 MHz copies needed, the second level buffers may not be necessary. Again, Intel has not performed any formal analysis on these parts.

4.3 Clock Generation for Asynchronous Systems

If the memory bus is not synchronized with the CPU or CPU-cache module, clock generation for the system is easier compared with the two configurations above. However, clock synchronization for the Pentium processor, 82496, and 82491, as well as the clocks for the MBC is still a concern. In order for the MBC to communicate properly with the CPU-Cache chip set, some synchronized clocks at 66 MHz are needed. Since the system is asynchronous to the CPU-Cache chip set, the number of synchronous MBC clock signals is less than the synchronous case. The examples in Section 5.0 illustrate how the synchronization is done. Since the system is asynchronous, one can use a different clock source for the CPU-Cache chip set from the rest of the system.

5.0 Pentium™ PROCESSOR WITH 256K 82496/82491 SECOND LEVEL CACHE CLOCK DISTRIBUTION DESIGN EXAMPLES

After a clock generation scheme is determined, careful analysis must be done on clock distribution to ensure

minimal skew and proper rise and fall times. Clock distribution is the connection between clock driver outputs and clock inputs of the components that need clocking. Preliminary analysis has been done on several of the drivers listed in Section 3.0. The following examples show in detail how to terminate transmission lines properly, tune clock traces to minimize board trace skew, and validate the usefulness of the drivers to the CPU-Cache chip set using models from the manufacturers. The examples have been done using preliminary or typical models for the devices involved. They are meant as an example of the process designers can use when selecting and routing a clock circuit. Although the examples only show the clock distribution for the CPU-Cache chip set, the same principles can be applied to distribution to the memory bus controller (MBC) and other parts.

5.1 Clock Routing for the 256K CPU-Cache Chip Set

Analysis for CPU-Cache chip set clock routing is done using first order input models for the three chips, shown in Figure 13. Specific to CLK inputs, the models shown are **typical models** based on on-going simulation efforts, and are subject to change. Refer to the Intel data books or contact Intel for the latest models (including minimum and maximum conditions). The models include package inductance, package capacitance, and input buffer capacitance of the clock pins.

3

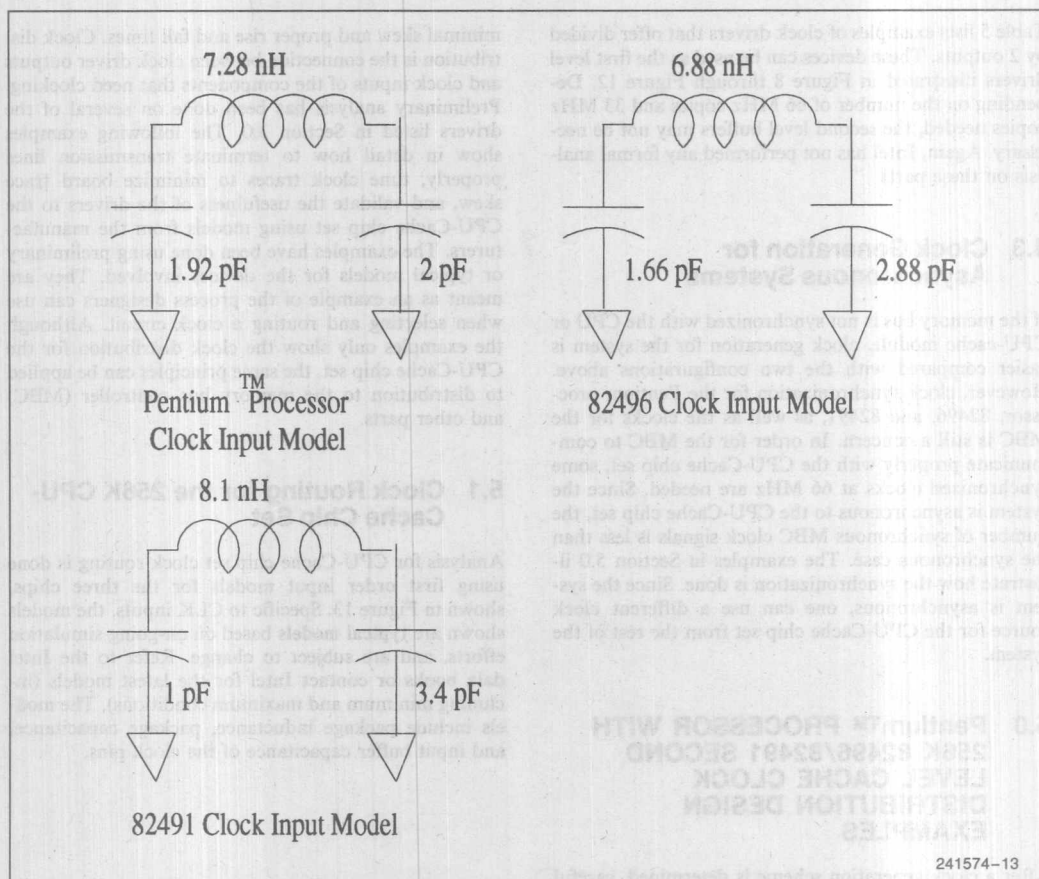


Figure 13. Pentium™ Processor, 82496 and 82491 Clock Input Models

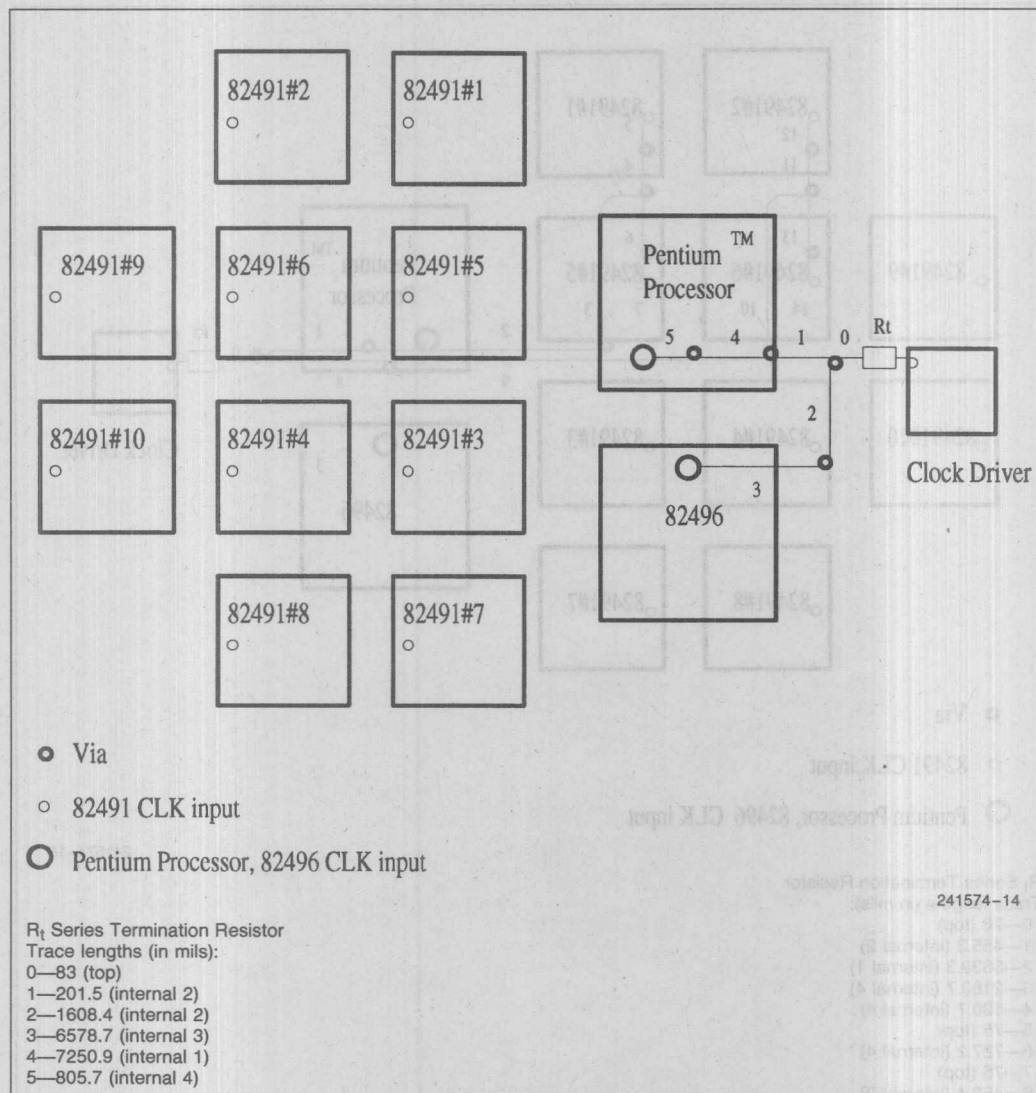


Figure 14. CLK0 Layout for 256K Chip Set with Parity

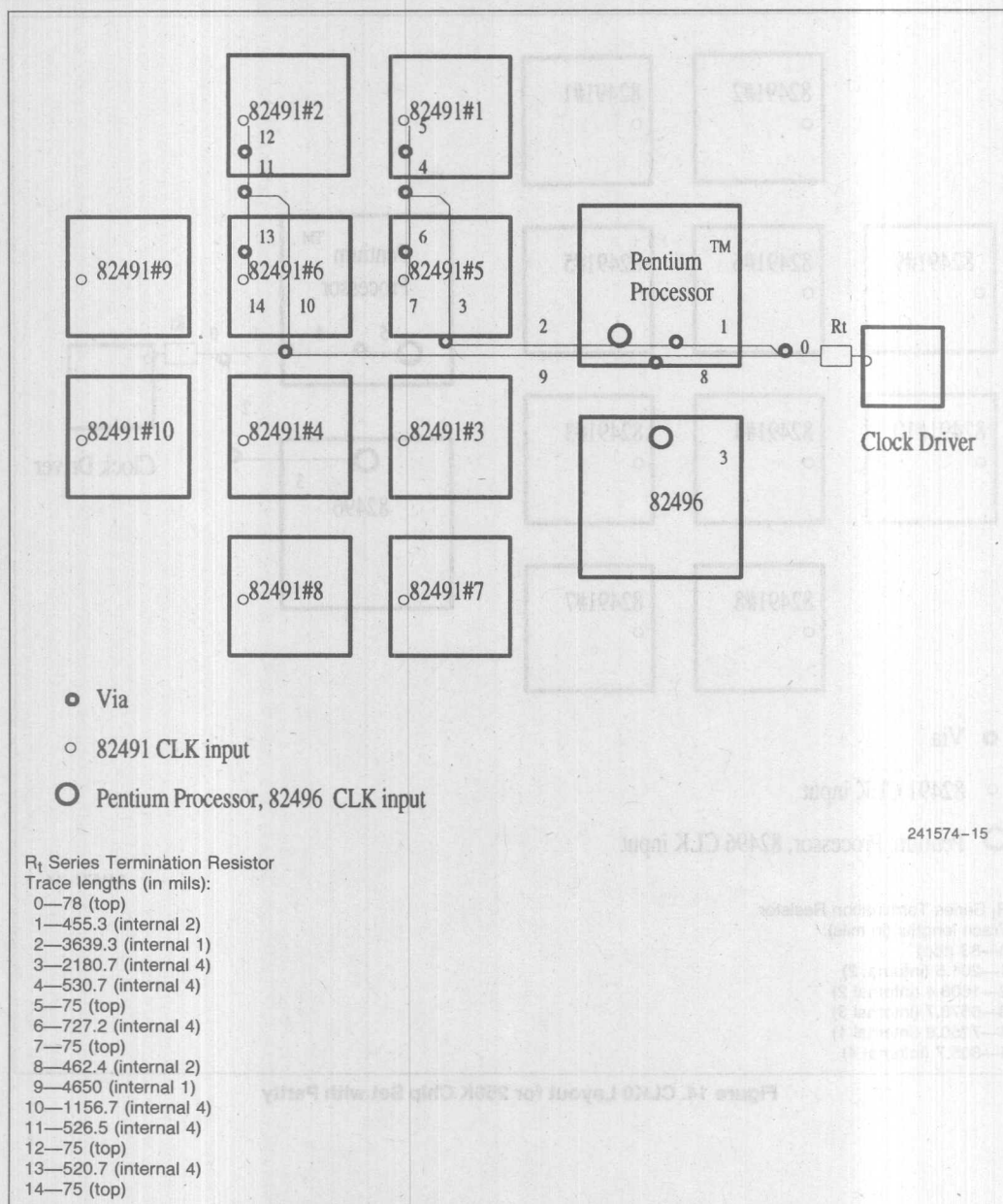


Figure 15. CLK1 Layout for 256K Chip Set with Parity

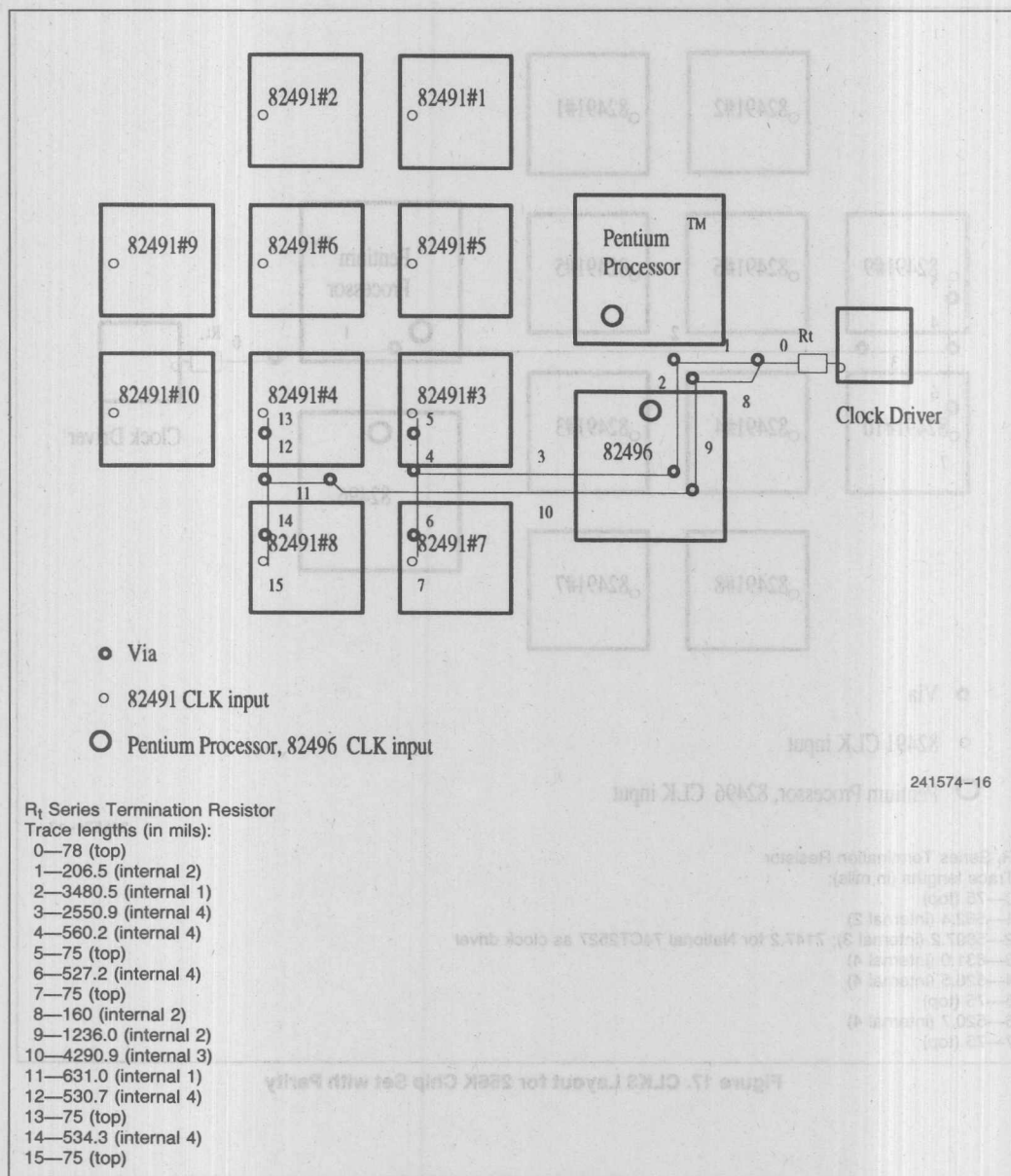


Figure 16. CLK2 Layout for 256K Chip Set with Parity

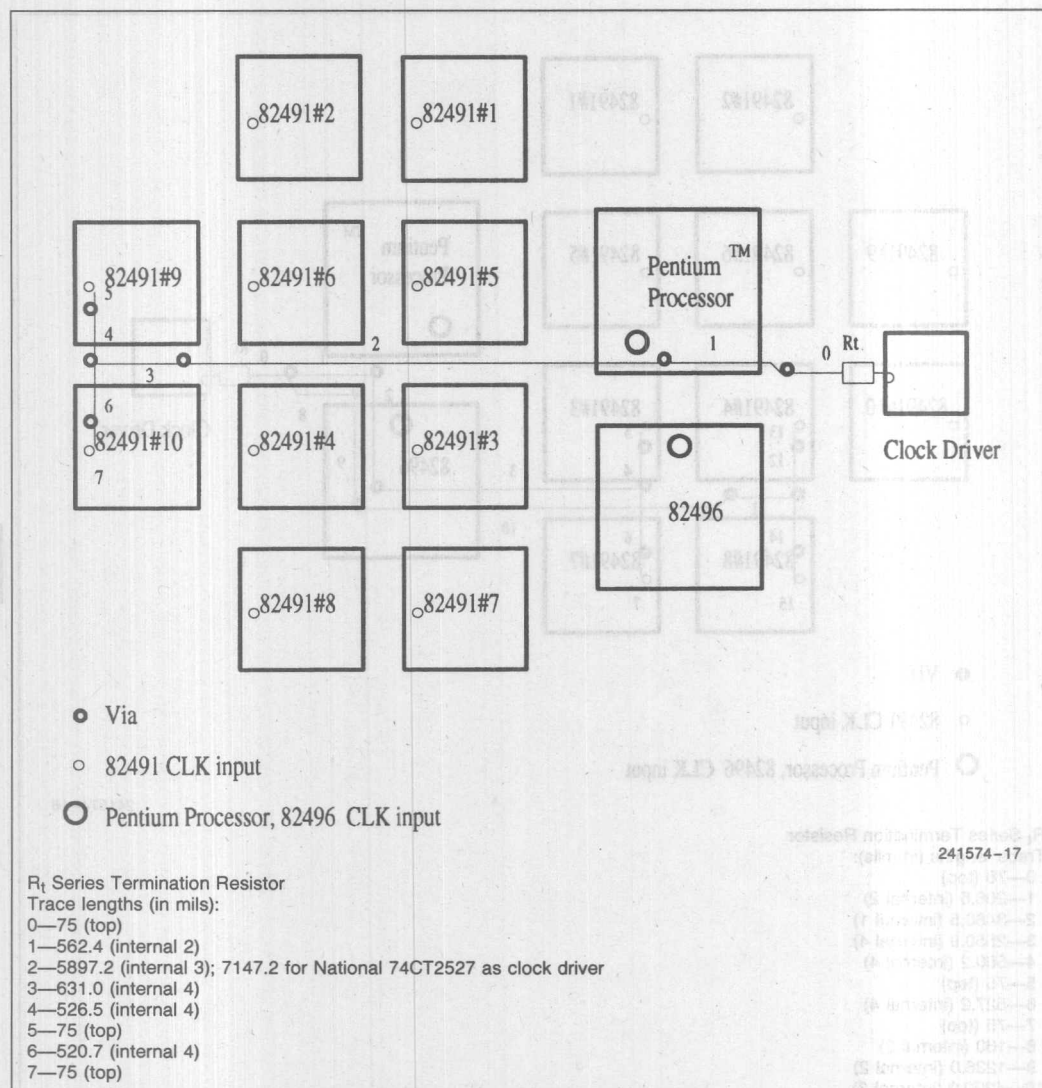


Figure 17. CLK3 Layout for 256K Chip Set with Parity

Figure 14 through Figure 17 show how clock signals are distributed from the clock driver to each component in the CPU-Cache chip set. Each clock line is tuned to minimize skew. Series termination is used on each line. Since the 82491, numbers 9 and 10, are parity chips, they are grouped onto the same driver output. Since the skew requirement between the Pentium processor and the 82496 is very tight (0.2 ns), they are also on the same driver output. All the loads on the same driver output can be tuned to have close to zero skew. Loads on different outputs, however, must contend with the output skew of the driver. CLK0 through CLK2 are laid out such that they branch off very close to the driver. For these lines, the transmission line characteristic impedance that the clock driver output sees can be treated as two resistors in parallel. In other words, the driver output sees half the impedance for CLK0, CLK1, and CLK2. The advantage of this scheme is that the value of the termination resistor is reduced dramatically. A smaller termination resistor helps faster rise and fall times. For CLK3, the branch off is at the end of the line, and thus the driver output sees the full characteristic impedance.

To achieve minimal skew, all the loads should be balanced, i.e., all the loads should be the same. For this design example, CLK1 and CLK2 have about twice the loads of CLK0 and CLK3. The load imbalance will add to the output skew quoted by manufacturers since on data sheets, manufacturers generally quote output skew for balanced loads. Since heavier loads see the transmitted signal later than lighter loads assuming the transmission lines are of the same length, traces for the lighter loads can be made longer to compensate for the discrepancy. CLK0 and CLK3 have longer traces from driver output to load than CLK1 and CLK2 traces. Since heavier loads (higher capacitance) have a longer rise time, and since for the CPU-Cache chip set skew measurements are taken at 0.8V, 1.5V, and 2.0V, to minimize skew at all free points, the termination resistors for CLK1 and CLK2 should be smaller than the termination resistors on CLK0. However, a smaller termination resistor than the value needed to perfectly terminate the line will result in a larger undershoot. When choosing termination values, it is a trade off among rise/fall times, skew, and undershoot.

When choosing a termination value, it is important to know the output impedance of the driver. For many TTL drivers, output rise impedance is different from output fall impedance. [Reference 3, Section 9] shows how to measure output impedance, or the driver manufacturer can be contacted for the information. Typically, output fall impedance is 5 Ω –10 Ω , and rise impedance is 5 Ω –50 Ω .

Figure 14 through Figure 17 are extensions to the layout topologies in the *Pentium™ Processor, 82496, and 82491 256K CPU-Cache Chip Set Layout Example*. The routing topologies 15–18 shown in the example route the clock signals from the Pentium processor, 82496, and all the 82491's to the outside. The layout examples shown in this application note (Figure 14 through Figure 17) take the layout all the way to the clock driver, complete with termination. Although in the example, clock signals are routed toward the bottom and in the examples here, the clock signals are routed toward the side, the same principles apply. If routing toward the bottom is preferred, the same layouts as illustrated in Figure 14 through Figure 17 can be used with little or no modification.

5.2 Analysis of Drivers Used in Examples

Output models for MC10H646, CGS74CT2527, GA1086, and VSL4485 are used to drive the clock network described in Section 5.1. The clock networks shown in Figure 14 through Figure 17 were used. The simulations assume no variation in characteristic impedance and propagation speed for the board traces. Fast and slow simulations were performed. Three sigma clock driver models are used when available. Board traces are assumed to have plus/minus 10% variation in characteristic impedance and propagational speed. Table 6 shows the range of trace characteristics. Slow simulations assume the highest operating temperature the drivers expect to see, and slow interconnect characteristics. Fast simulations assume operating temperature to be zero and fast interconnect characteristics.

Table 6. Interconnect Characteristics

Corner	Trace Type	Z0(1) (Ω)	TD(2) (ns/ft)
Slow	Inner	58.5	2.41
Fast	Inner	71.5	1.85
Slow	Surface	72	2.05
Fast	Surface	108	1.35

NOTES:

1. Characteristic Impedance
2. Propagational Speed

Since simulation can only account for skew due to board trace and load imbalance, total skew is assumed to be the sum of the worst case output skew published by driver manufacturers and skew from simulation. Skew from simulation is derived by using identical driver models for each driver output, thus assuming zero

output skew. The board traces and termination resistances are tuned with 0.5 ns output skew in mind which leaves 0.2 ns for trace skew. For TriQuint's GA1086, the output skew is 0.25 ns; thus, there is a larger window for trace skew. Table 7 summarizes simulation results of the tightest parameters for each driver. All of the drivers can meet the 4 ns minimum high and low times easily. Most clock drivers guarantee a 45/55 duty cycle, which exceeds Intel's requirement.

Series termination resistors are chosen to minimize skew and undershoot. To achieve similar rise time for each load, termination resistance values are smaller for heavier loaded lines such as CLK1 and CLK2 compared to the resistance values for CLK0. Since CLK3 splits off at the end of the line, its termination value is about twice as CLK0's. Table 8 lists the termination values for each line and for each driver. Waveforms for each driver are attached. Notice the signals at the CLK

input for each load is relatively clean whereas the signals at the driver side are not. Since the clock signals are only important to the component receiving the signal, how dirty the signal is at the driver end is not important, providing that the signal does not cause any damage or other ill effects on the driver.

Figures attached in this section show some waveforms from the simulations. V(201) is the voltage at the Pentium processor clock input, V(202) is the voltage at the 82496 clock input, and V(213), V(214), V(217), and V(218) are voltages at the 82491 clock inputs for the 82491s on CLK1 line. For 74CT2527, V(8) is the voltage at driver output, V(100) is the voltage at the junction of the series termination resistor and the beginning of board trace. For 10H646, V(9) is the voltage at driver output, V(20) is the voltage at the junction of the series termination resistor and the beginning of board trace.

Table 7. Compilation of Simulation Data

Mfr.	Clock Driver	Worst Skew P5-C5C (ns)(1)			Worst Skew C8C-Others (ns)			Worst Skew C8C (No Parity) (ns)(2)			Undershoot (-mV)(3)			Tr/Tf (ns)(4)		
		Slow	Fast	Spec	Slow	Fast	Spec	Slow	Fast	Spec	Slow	Fast	Spec	Slow	Fast	Spec
Motorola	10H646	0.021	0.023	0.2	0.65	0.67	0.7	0.65	0.67	0.7	468	816	1600	0.90/1.13	0.74/0.67	1.5/1.5
National	74CT2527	0.0071	(5)	0.2	0.67	(5)	0.7	0.61	(5)	0.7	285	(5)	1600	1.14/0.42	(5)	1.5/1.5
TriQuint	GA1086	0	0	0.2	0.55	0.45	0.7	0.45	0.45	0.7	150	400	1600	0.9/1.9	0.6/1.2	1.5/1.5
Vitesse	VSL4485	0.02	0.05	0.2	0.7	0.57	0.7	0.66	0.57	0.7	275	800	1600	0.95/0.78	0.78/0.6	1.5/1.5

NOTES:

1. All Skews are worst case numbers
2. Not using the parity chips
3. Worst Undershoot of all the CLK nodes
4. Slowest rise and fall times of all the CLK nodes
5. Only typical model at 25°C is available. Thus, only simulation performed is with slow interconnect corner
6. Simulation done on driver slow corner. Device specification for t_f is 1.4 ns worst case. Device was still under development when simulation was done. Please contact TriQuint for more information.

Clock distribution method for the memory bus controller (MBC) is very similar to that of the chip set. When distributing clocks for the MBC, be sure to load each driver output with similar loads as for the chip set, and route clock traces with similar lengths as for the chip set. For example, CLK1 and CLK2 have an aggregate load of about 20 pF, and the total clock trace length is about 7" from driver output to a load. To minimize the clock skew of the MBC clock from loads on CLK1 and CLK2 lines, the clock lines should fan out 2.9 pF per inch. Also, be sure to terminate the line properly. It is important to keep the loading similar to the loadings on clock lines of the chip set if skew is to be kept close to 0.7 ns. Adjusting trace lengths and termination resistance can compensate for load imbalance to a degree, but not perfectly and not always.

Simulations results provided here are based on best available models at the time. Some models were for parts still under development at the time of simulation. Therefore, the simulation results are subject to change.

Table 8. Series Termination Resistor Values for Each Line

Manufacturer	Clock	CLK Line	R _t (Ω)
Motorola	10H646	CLK0	26
		CLK1	20
		CLK2	20
		CLK3	59
National	74CT2527	CLK0	20
		CLK1	15
		CLK2	15
		CLK3	45
TriQuint	GA1086	CLK0	30
		CLK1	30
		CLK2	30
		CLK3	50
Vitesse	VSL4485	CLK0	32
		CLK1	23
		CLK2	23
		CLK3	48

Figure 18. Motorola 646 Waveform

Motorola 646 clock driver simulation, clk0 slow
 Date/Time run: 03/31/92 11:04:34
 Temperature: 90.0

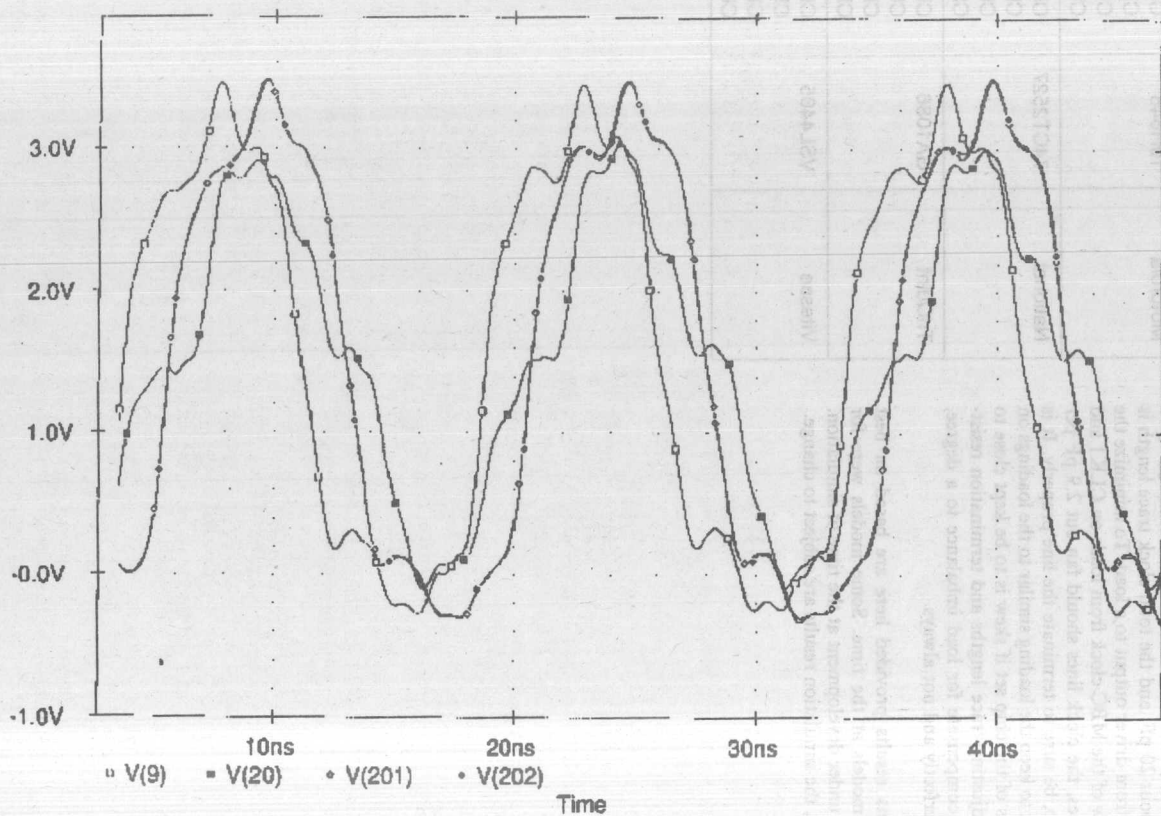
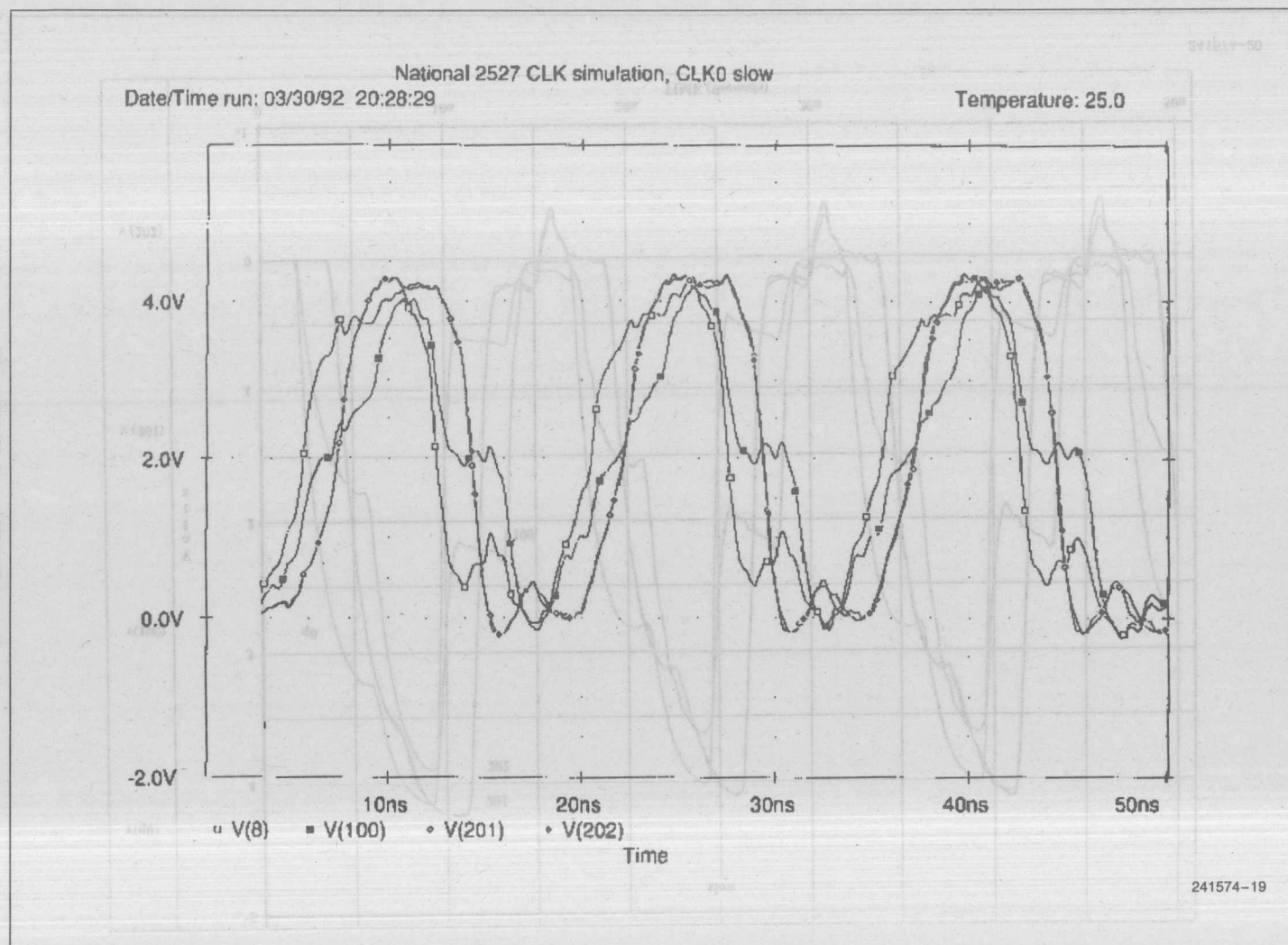
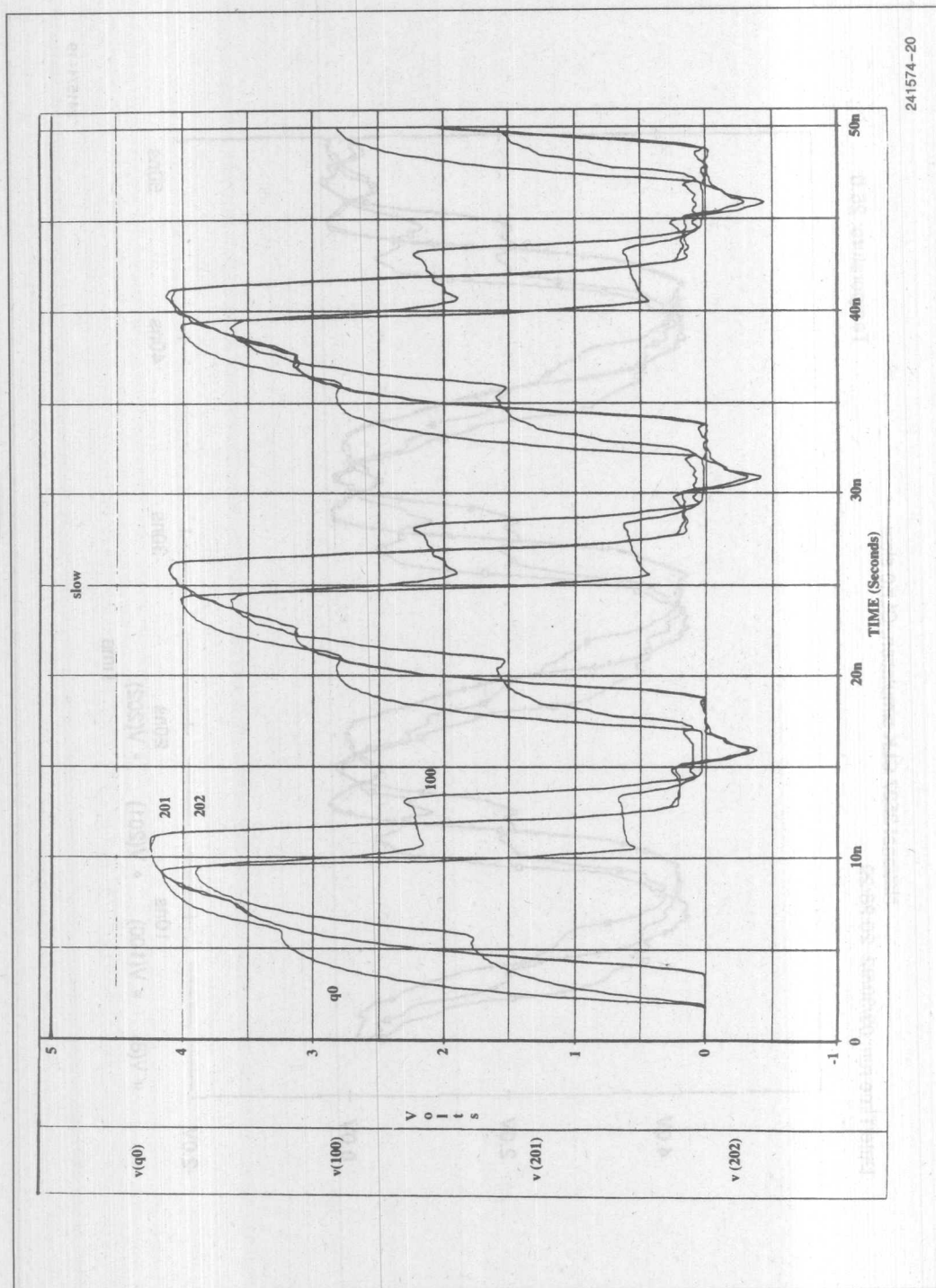


Figure 19. National Waveform





241574-20

Figure 20. Vitesse (Slow) Waveform

241574-21

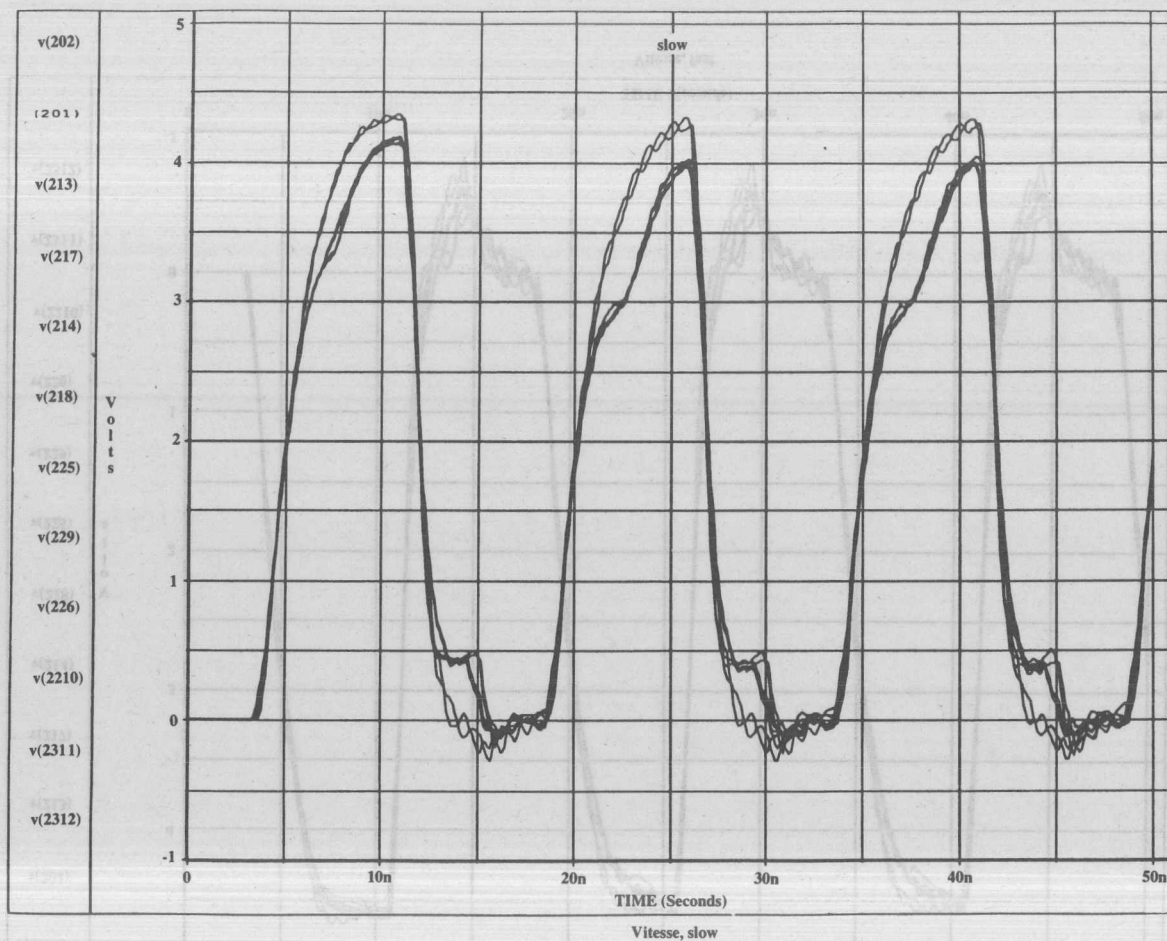


Figure 21. Vitesse (Slow) Waveform (Continued)

PRELIMINARY

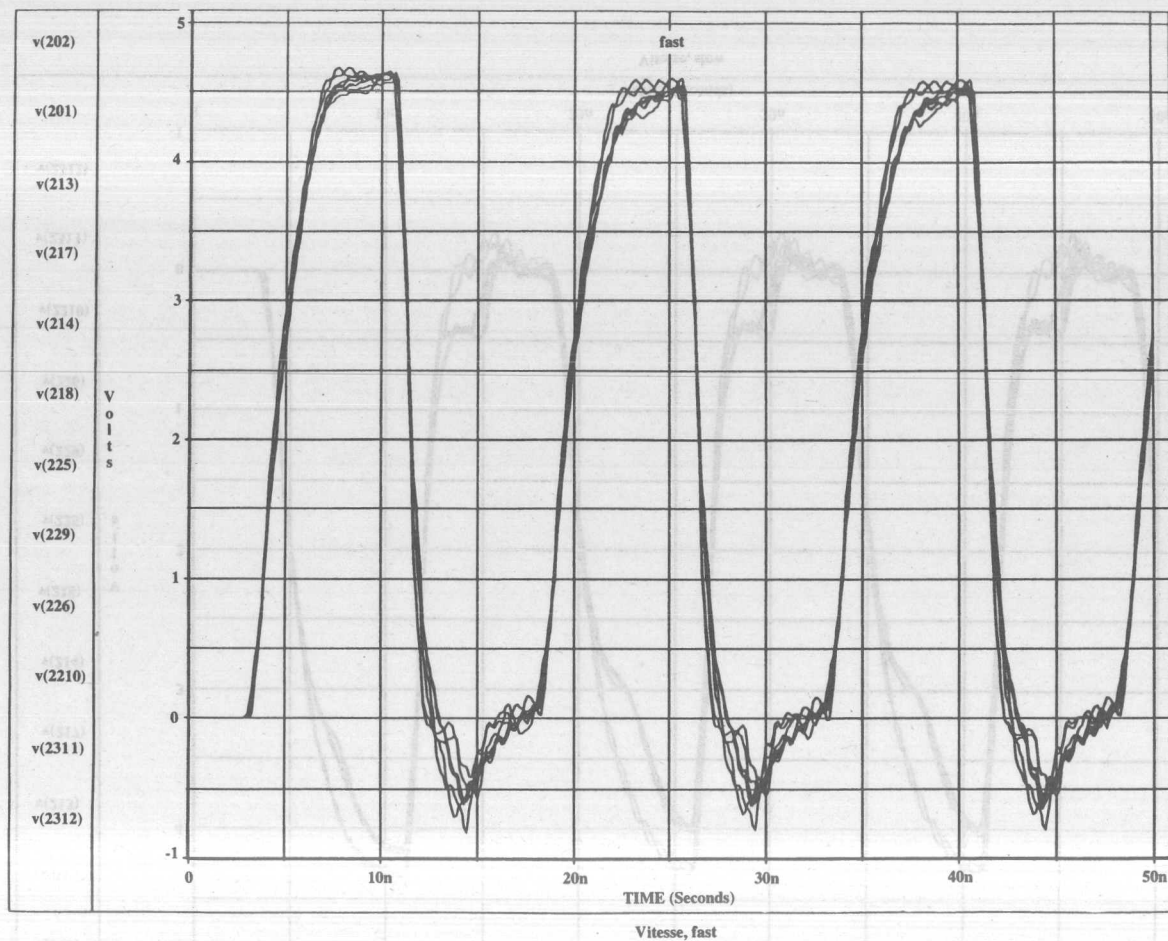


Figure 22. Vitesse (Fast) Waveform

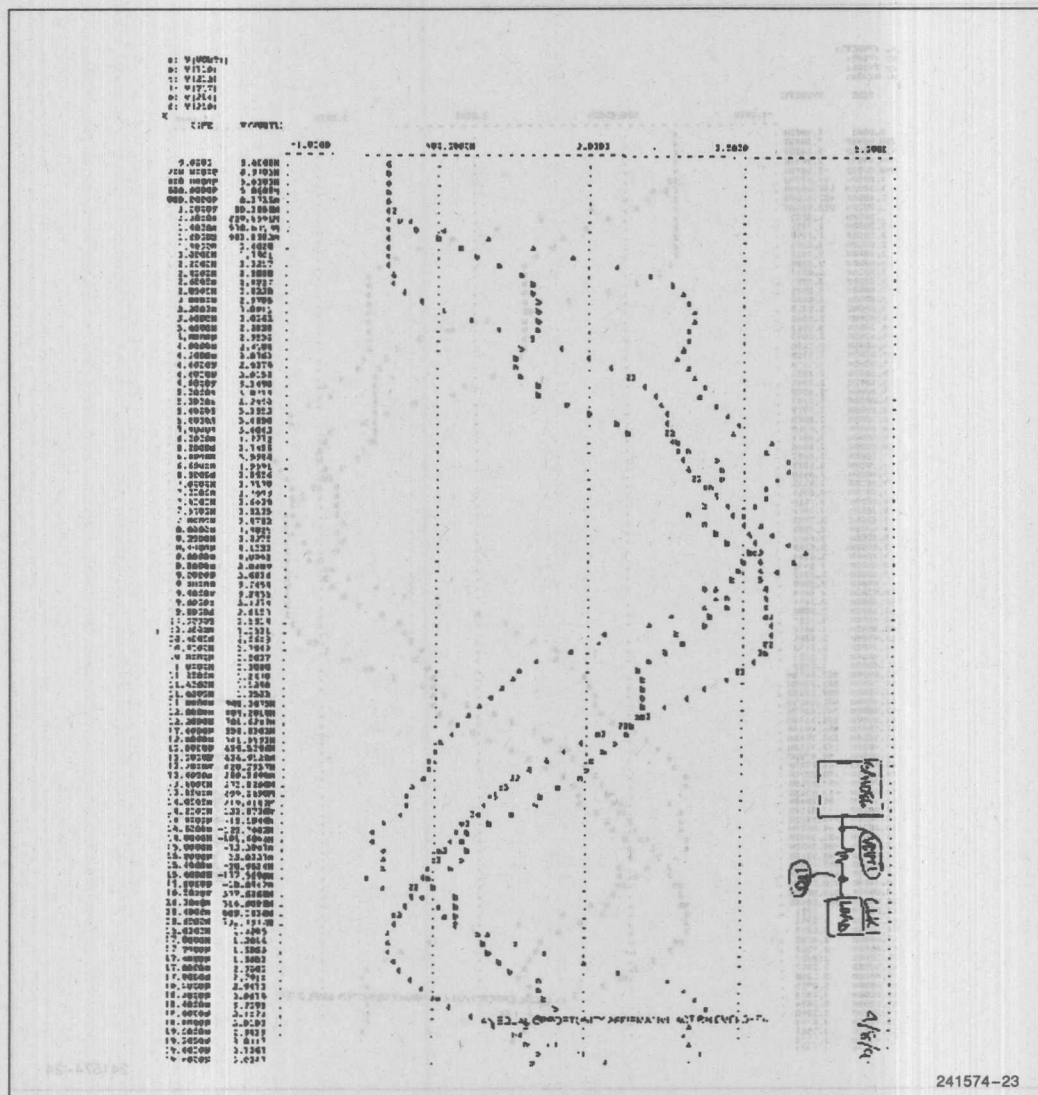


Figure 23. Triquint Waveform

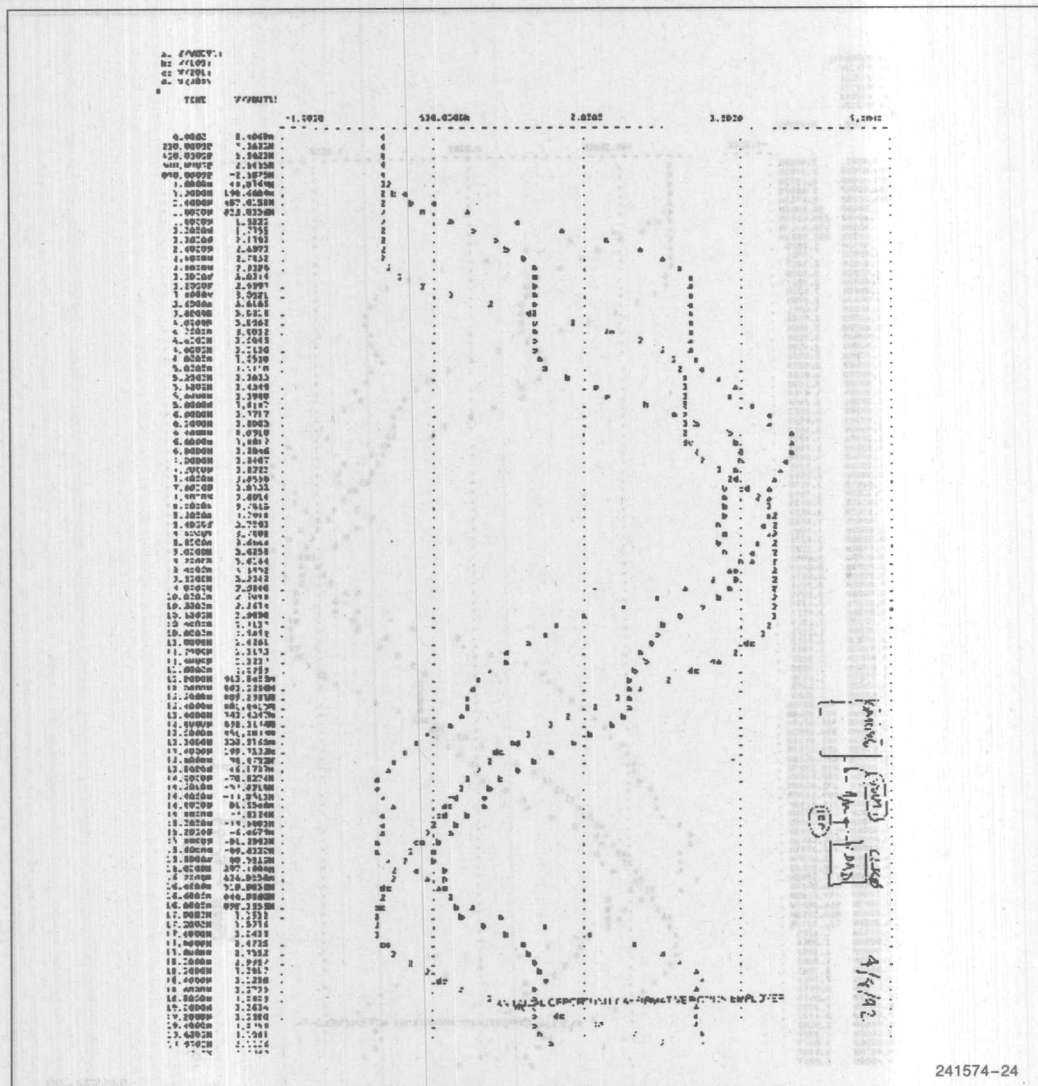


Figure 24. Triquint Waveform (Continued)

6.0 Pentium™ PROCESSOR WITH 512K 82496/82491 SECOND LEVEL CACHE CLOCK DISTRIBUTION ISSUES

Clock distribution for 512K CPU-Cache chip set can be done in the same way as for the 256K chip set. Since there are more SRAM chips for the 512K cache, there are more loads that need clocking. Including parity, there are 18 82491s. Once again, the same principles apply. Keep the driver loading as close to balanced as possible. Tune traces and adjust termination resistance so that skew is minimized.

7.0 CLOCK DISTRIBUTION FOR THE Pentium™ PROCESSOR WITH OTHER SECOND LEVEL CACHES

The Pentium processor can be used with cache configurations other than with the 82496 and 82491, as well as, without a second level cache. With other caches, the first thing that must be done is to decide how much skew is tolerable. Then, decide on which clock driver to use and carefully layout clock signals for distribution. If skew requirements do not exceed the CPU-Cache chip set requirements, the same drivers and the same distribution can be used. Design examples in Section 5.0 serve as a guide to how to distribute clocks for Pentium processor systems with tight skew.

If the Pentium processor is used without a second level cache, and only a small number of 66 MHz signals are needed, there are a few more options for clock drivers. For example, Motorola's 88915 has one 2x output that can run to maximum 70 MHz. Texas Instruments has the ABT337, 338, and 339 that can provide four copies of 66 MHz signals.

8.0 SUMMARY

At high speeds, clock synchronization becomes a difficult problem. Clock traces must be treated as transmission lines. Proper termination must be given to the lines to ensure good signal quality. The Pentium processor, with operating frequencies of 60 MHz and 66 MHz, has tight clock requirements. Together with the 82496 Cache Controller and 82491 Cache SRAM, the CPU-Cache chip set must be synchronized with minimal skew.

For the Pentium processor clocking, the most critical parameters are skew and rise and fall times. Depending

on the memory interface to the CPU-Cache chip set, there are many ways of generating multiple copies of clock signals.

Fully synchronous designs need to route 66 MHz only, but with minimal skew for all of them. Divided synchronous designs require both 66 MHz and 33 MHz signals. Asynchronous designs need to worry about the CPU-Cache chip set clock generation and distribution as well as the MBC.

Several clock drivers have been analyzed in detail with carefully tuned clock routing and the proper termination such that the clock signals transmitted to the Pentium processor, 82496, and 82491 meet all the timing requirements of the Intel chip set parts. Loading on a clock driver should be as balanced as possible. Clock traces should have equivalent length from driver output to load. The clock lines should be terminated properly to minimize reflections.

The same design principles used in the 256K CPU-Cache chip set clocking example can be applied to other CPU-cache configurations, or to a cacheless interface.

This application note has listed a number of devices from several different manufacturers. The purpose of this list is to supply a starting point for finding a clocking solution that meets each system's specific requirements. The lists provided are not meant as an endorsement or guarantee of the devices listed. In addition, these lists are not a complete listing of devices. These or other manufacturers may offer additional devices that meet the clock specifications for the Pentium processor.

9.0 REFERENCES

1. Intel Corporation, *Pentium™ Processor User's Manual*, Order Number: 241563.
2. Intel Corporation, *Designing with the Pentium™ Processor, 82496, and 82491 256K CPU-Cache Chip Set*, Order Number: 241576.
3. Jolly, Rich, *Clock Design in 50 MHz Intel486™ Systems*, Application Note AP-453, 1991, Intel Corporation, Santa Clara, CA.
4. Blood, William R., Jr., *MECL System Design Handbook*, 1988, Motorola Inc.
5. Hanke, Chris and Tharalson, Gary, *Low Skew Clock Drivers and their System Design Considerations*, Application Note AP-1091, Motorola Inc., 1990.

APPENDIX A CLOCK DRIVER MANUFACTURERS

The following is a list of contacts for the clock driver manufacturers listed in this application note. It is not meant to be an exhaustive list of all possible solutions. It is meant as a starting point for system designers to assist in finding a clock solution that meets their system requirements.

AMCC

United States:

Headquarters

6195 Lusk Boulevard

San Diego, CA 92121-2793

Ph: 619-450-9333 or 800-PLL-AMCC (755-2622)

FAX: 619-450-9885

Europe:

Alpha Electronics

Basingstoke, RG24OPF, U.K.

Ph: 011/44-256-843166

Japan:

Teksel Co., Ltd.

Kawasaki 213

Tokyo, Japan

Ph: 011/81-448127430

Israel:

EIM

Petach Tiqva, Israel

Ph: 011/972-3-9233257

AT&T Microelectronics

AT&T Customer Response Center

Ph: 800-372-2447 x773

Danny George

555 Union Blvd.

Allentown, PA 18103

50N2G2100

Ph: 215-439-6697

Cypress

Sean Dingman

3901 N. 1st St.

San Jose, CA 95134

408-943-2743

ICS

Bruce Rogers

Technical Marketing Manager

2626 Van Buren Ave.

P.O. Box 968

Valley Forge, PA 19482

215-666-1900

Intel PLD BU International Contact List

United States:

John Van Sack

Intel Corporation

FM4-42

1900 Prairie City Road

Folsom, CA 95630

Ph: (916) 356-3964

FAX: (916) 356-6949

Europe:

Tony O'Sullivan

Intel Corporation GmbH

Dornacher Str. 1

Postfach 213

D-8016 Feldkirchen/Munchen

Germany

Ph: (49) 89/90992-340

FAX: (49)89/9043948

Japan:

Norikazu Aoki

5-6 Tokodia, Tsukuba-shi

Ibaraki-Ken 300-26

Japan

Ph: 0298-47-0721

FAX: 0298-47-8819

APAC:

Eric Chan

Intel Technology SDN BHD

Bayan Lepas Free Trade Zone,

Box 121

11900 Penang

Malaysia

Ph: 604-820-7271

FAX: 604-836-405

Motorola Inc.

Todd Pearson
Motorola Inc
2200 W. Broadway Rd.
Mesa, Arizona 85202
USA
Ph: (602) 962-3410

Masanori Matsubara
Nippon Motorola LTD
3-20-1, Minami-Azabu
Minato Ku, Tokyo 106
Japan
Ph: 81-33-280-8383

Axel Krepil
Motorola GMBH
Schatzbogen 7
8000 Munchen 81
Germany
Ph: 49-89-92103-167

Derek Leung
Motorola Hong Kong LTD
Silicon Harbour Center
2 Dai King Street
Taipo Industrial Estate
Taipo N. T. Hong Kong
Ph: 852-666-8194

National Semiconductor

National Semiconductor
Santa Clara, CA
Tony Ochoa
Ph: 408-721-6804
Ph: 800-272-9959

Pioneer Semiconductor

Joe Kraus
2343 Bering Dr.
San Jose, CA 95131
Ph: 408-435-0800
FAX: 408-435-1100

Texas Instruments
United States:

Steve Plote
Program Manager
CLOCK DRIVERS
8330 LBJ Freeway, Center 3
P.O. Box 655303
Dallas, Texas 75265
Ph: 214-997-5214

Brett Clark
Applications Engineer
Ph: 903-868-5836

Japan:

Mich Komatsu
Texas Instruments Japan LTD.
M.S. Shibaura Bldg. 13-23
Shibaura 4-Chome
Minato-ku Tokyo, 108 Japan
Ph: 033-769-8717

Asia Pacific Region:

Eric Wey
Texas Instruments Taiwan LTD.
Taipei Branch
10F Bank Tower, 205 Tung Hua N.
Taipei, Taiwan ROC
Ph: 886-2-713-9311

Europe:

Lothar Katz
Texas Instruments
8050 Freising, Fed. Rep. of Ger.
Deutschland GMBH
Haggertystr. 1
Ph: 49-816-80314

TriQuint Semiconductor
United States:

Marketing, Sunil Sanghavi
(408) 982-0900 x142, FAX (408) 982-0222
Western Sales, Mark Wu
(408) 982-0900 x113, FAX (408) 982-0222
Central Sales, John Watson
(214) 422-2532, FAX (214) 423-4947
East Sales, Mike Zyla
(215) 493-6944, FAX (215) 493-7418
International, Mike Kilgore
(503) 644-3535 x228, FAX (503) 644-3198

Europe - GiGA A/S

Fin Helmer, President
45-4-343-1588, FAX 45-4-343-5967

Japan - Japan Macnica Corp.

Shin Ishikawa, Product Manager
045-939-9140, FAX 045-939-6141

Vitesse Semiconductor

United States:

Corporate Headquarters
Vitesse Semiconductor Corporation
741 Calle Plano
Camarillo, CA 93012
Ph: 805-388-7501
FAX: 805-388-7565

Europe:

Thomson Composants Micronodes
Route Departementale 128 B.P. 48
91401 Orsay Cedex France
Ph: 33-1-60-19-7000
FAX: 33-1-60-19-7140

Japan:

H.Y. Associates Co., LTD.
3-1-10, Sekimachikita, Nerima-Ku
Tokyo, 177 Japan
Ph: 81-33-929-7111
FAX: 81-33-928-0301

Korea:

Beaver International, Inc.
3601 Deauville Court
Calabasas, CA 91302
Ph: 818-591-0356
FAX: 818-591-0753

Taiwan:

Tamarack Microelectronics
16 Fl., No. 1, Fu-Hsing N. Road
Taipei, Taiwan, ROC
Ph: 886-2-772-7400
FAX: 886-2-776-0545

APPLICATION NOTE

Pentium™ PROCESSOR THERMAL DESIGN GUIDELINES REV. 2.0

3

DAN McCUTCHAN

November 1993

PRELIMINARY

Order Number: 241575-001

3-169

Pentium™ PROCESSOR THERMAL DESIGN GUIDELINES REV. 2.0

CONTENTS	PAGE	CONTENTS	PAGE
1.0 INTRODUCTION	3-171	5.0 DESIGNING FOR THERMAL PERFORMANCE	3-175
1.1 Document Goal	3-171	5.1 Heat Sinks	3-176
2.0 IMPORTANCE OF THERMAL MANAGEMENT	3-171	5.2 Airflow	3-180
3.0 Pentium™ PROCESSOR POWER SPECIFICATIONS	3-172	5.3 Fans	3-180
4.0 THERMAL PARAMETERS	3-172	5.4 Thermal Performance Validation	3-180
4.1 Ambient Temperature	3-172	6.0 CONCLUSION	3-180
4.2 Case Temperature	3-173	APPENDIX A	3-181
4.3 Junction Temperature	3-173	APPENDIX B	3-195
4.4 Thermal Resistance	3-174		

1.0 INTRODUCTION

In a system environment, the Pentium™ processor's temperature is a function of both the system and component thermal characteristics. The system level thermal constraints imposed on the package are local ambient temperature and thermal conductivity (i.e., airflow over the device). The Pentium processor thermal characteristics depend on the package (size and material), the type of interconnection to the printed circuit board (PCB), the presence of a heat sink, and the thermal conductivity and the power density of the PCB.

All of these parameters are aggravated by the continued push of technology to increase the operating speeds and the packaging density. As operating frequencies increase and packaging size decreases the power density increases and the heat sink size and airflow become more constrained. The result is an increased importance on system design to ensure that thermal design requirements are met for each component in the system.

In addition to heat sinks and fans, there are other solutions for cooling integrated circuit devices. A few of these solutions are: fan mounted on heat sink, heat pipes, thermoelectric (peltier) cooling, liquid cooling, etc. While these alternatives are capable of dissipating additional heat, they have disadvantages in terms of system cost, complexity, reliability, and efficiency. These techniques are more expensive than a passive heat sink and fan. The introduction of active devices can also decrease reliability. Finally, the power efficiency of some of these techniques is poor, and gets worse as the amount of power being dissipated increases. Despite these disadvantages, each of these solutions may be the right one for particular system implementations.

However, for the purpose of this application note, Intel has focused its efforts on describing solutions using passive heat sinks and fans.

1.1 Document Goal

The goal of this document is to provide thermal performance information for the Pentium processor and recommendations for meeting the thermal requirements imposed on systems. This application note attempts to provide an understanding of the thermal characteristics of the Pentium processor and some examples of how the thermal requirements can be met.

2.0 IMPORTANCE OF THERMAL MANAGEMENT

Thermal management of an electronic system encompasses all of the thermal processes and technologies that must be employed to remove and transfer heat from individual components to the system's thermal sink in a controlled manner.

The objective of thermal management is to ensure that the temperature of all components is maintained within functional and absolute maximum limits. The functional temperature limit is the range within which the electrical circuits can be expected to meet their specified performance requirements. Operation outside the functional limit can degrade system performance or cause logic errors. The absolute maximum temperature limit is the highest temperature that a portion of the component may be safely exposed. Temperatures exceeding the limit can cause physical destruction or may result in irreversible changes in operating characteristics. Higher temperatures result in earlier failure of the devices in the system. Every 10°C rise above the operating range means a halving of the mean time between failures.

3.0 Pentium™ PROCESSOR POWER SPECIFICATIONS

The Pentium processor's power dissipation and case temperature specs for 60 MHz and 66 MHz are shown in Table 1.

To ensure functionality and reliability of the Pentium processor, maximum device junction temperature must remain below 90°C. Considering the power dissipation levels and typical ambient environments of 40°C to 45°C, the Pentium processor's junction temperatures cannot be maintained below 90°C without additional thermal enhancement to dissipate the heat generated by this level of power consumption.

The thermal characterization data described in Table 2 illustrates that both a heat sink and airflow are needed. The size of heat sink and the amount of airflow are interrelated and can be traded off against each other. For example, an increase in heat sink size decreases the amount of airflow required. In a typical system, heat sink size is limited by board layout, spacing, and component placement. Airflow is limited by the size and number of fans along with their placement in relation to the components and the airflow channels. In addition, acoustic noise constraints may limit the size or types of fans limiting the airflow.

To develop a reliable thermal solution, all of the above variables must be considered. Thermal characterization and simulation should be carried out at the entire sys-

tem level accounting for the thermal requirements of each component.

4.0 THERMAL PARAMETERS

Component power dissipation results in a rise in temperature relative to the temperature of a reference point. The amount of rise in temperature depends on the net thermal resistance between the junction and the reference point. Thermal resistance is the key factor in determining the power handling capability of any electronic package.

Thermal resistance from junction to case (θ_{JC}), and from junction to ambient (θ_{JA}) are the two most often specified thermal parameters for integrated circuit packages.

4.1 Ambient Temperature

Ambient temperature is the temperature of the undistributed ambient air surrounding the package. Denoted T_A , ambient temperature is usually measured at a specified distance away from the package. In the laboratory test environment, ambient temperature is measured 12 inches upstream from the package under investigation. In a system environment, ambient temperature is the temperature of the air upstream to the package and in its close vicinity.

Table 1. Pentium™ Processor Power Dissipation

	Package Type	Total Pins	Pin Array	Package Size	Power (Typical)	Power (Max)	Max Case Temp (°C)
Pentium Processor 60 MHz	PGA	273	21 x 21	2.16" x 2.16"	11.9W	14.6W	80
Pentium Processor 66 MHz	PGA	273	21 x 21	2.16" x 2.16"	13W	16W	70

4.2 Case Temperature

Case temperature, denoted T_C , is measured at the center of the top surface (on top of the heat spreader, see Figure 1) of the package, typically the hottest point on the package case. Special care is required when measuring the case temperature to ensure an accurate temperature measurement. Thermocouples are often used to measure T_C . Before any temperature measurements, the thermocouples have to be calibrated. When measuring the temperature of a surface which is at a different temperature from the surrounding ambient air, errors could be introduced in the measurements. The measurement errors could be due to having a poor thermal contact between the thermocouple junction and the surface, heat loss by radiation or by conduction through thermocouple leads. To minimize the measurement errors, it is recommended to use the following approach:

- Use 36 gauge or finer diameter K, T, or J type thermocouples. The laboratory testing was done using a thermocouple made by Omega (part number: 5TC-TTK-36-36).
- Attach the thermocouple bead or junction to the center of the package top surface using high thermal conductivity cements. The laboratory testing was done by using Omega Bond (part number: OB-100).
- The thermocouple should be attached at a 90° angle as shown in Figure 1. When a heat sink is attached a hole (no larger than 0.15") should be drilled through the heat sink to allow probing the center of the package as shown in Figure 1.

- If the case temperature is measured with a heat sink attached to the package, drill a hole through the heat sink to route the thermocouple wire out.

4.3 Junction Temperature

Junction temperature, denoted T_J , is the average temperature of the die within the package.

The junction temperature for a given junction-to-ambient thermal resistance, power dissipation, and ambient temperature is given by the following formula:

$$T_J = P_D \cdot \theta_{JA} + T_A$$

If a heat sink with thermal resistance of θ_{SA} (sink-to-ambient) is used, then the thermal resistance from the junction-to-case, θ_{JC} , is given by the following formula:

$$T_J = P_D \cdot (\theta_{JC} + \theta_{CS} + \theta_{SA}) + T_A$$

where:

θ_{CS} is the thermal resistance from the component (case) to the heat sink.

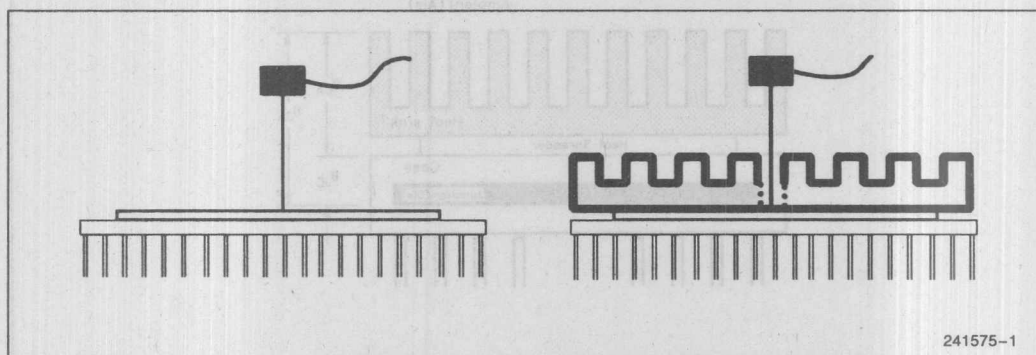


Figure 1. Thermocouple Attachment

Thermal resistance (Figure 2) values for junction-to-ambient, θ_{JA} and junction-to-case, θ_{JC} , are used as measures of IC package thermal performance. θ_{JC} is a measure of the package's internal thermal resistance along the major heat flow path from silicon die to package exterior. This value is strongly dependent on the material, thermal conductivity, and geometry of the package. θ_{JC} values also depend on the location of the reference point (in this case center of the package top surface), the external cooling configurations and the heat flow paths from the package to the ambient. For example, if a heat sink is attached to the package top surface or more heat is pulled into the board through the pins, the θ_{JC} values measured with reference to the center of the package top surface will change. θ_{JA} values include not only internal thermal resistance, but also the radiative and convective thermal resistance from the package exterior to ambient air. θ_{JA} values depend on the material, thermal conductivity, and geometry of the package and also on ambient conditions such as airflow rates and coolant physical properties.

In order to obtain thermal resistance values, junction temperature is measured using the temperature sensitive parameter (TSP) method. With this method, special design thermal test structures are used which are approximately the same size as the Pentium processor die. The test structure consists of resistors and diodes.

power dissipation and thereby heat up the package. Diodes, which are located at the center of the thermal test die, are used to measure the die temperature. The measurements are carried out in a wind tunnel environment. The air flow rate and the ambient temperature are measured 12 inches away from the package in the upstream air.

The parameters are defined by the following relationships:

$$\theta_{JA} = (T_J - T_A)/P_D$$

$$\theta_{JC} = (T_J - T_C)/P_D$$

$$\theta_{JA} = \theta_{JC} + \theta_{CA}$$

where:

θ_{JA} = junction-to-ambient thermal resistance
($^{\circ}\text{C}/\text{W}$)

θ_{JC} = junction-to-case thermal resistance ($^{\circ}\text{C}/\text{W}$)

θ_{CA} = case-to-ambient thermal resistance ($^{\circ}\text{C}/\text{W}$)

T_J = average die (junction) temperature ($^{\circ}\text{C}$)

T_C = case temperature at a pre-defined location ($^{\circ}\text{C}$)

T_A = ambient temperature ($^{\circ}\text{C}$)

P_D = device power dissipation (W)

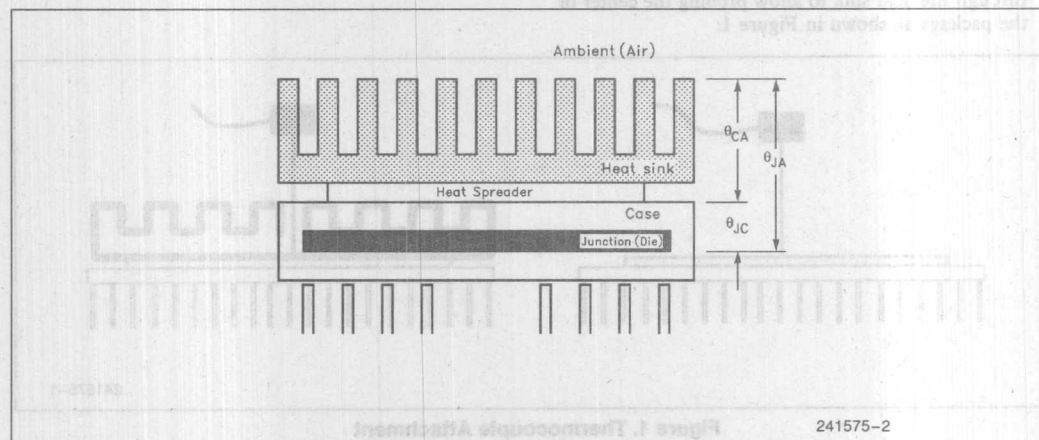


Figure 2. Thermal Resistance Parameters

Table 2 lists the junction-to-case and case-to-ambient thermal resistances for the Pentium processor (with and without a heat sink).

Table 2. Thermal Characterization Data

	θ_{JC}	θ_{CA} **vs Airflow (ft/min.)					
		0	200	400	600	800	1000
With 0.25" Heat Sink	0.6	8.3	5.4	3.5	2.6	2.1	1.8
With 0.35" Heat Sink	0.6	7.4	4.5	3.0	2.2	1.8	1.6
With 0.65" Heat Sink	0.6	5.9	3.0	1.9	1.5	1.2	1.1
Without Heat Sink	1.2	10.5	7.9	5.5	3.8	2.8	2.4

NOTE:

Heat Sink: 2.1 in² base, omni-directional pin Al heat sink with 0.050 in. pin width, 0.143 in. pin-to-pin center spacing and 0.150 in. base thickness. Heat sinks are attached to the package with a 2 to 4 mil thick layer of typical thermal grease. The thermal conductivity of this grease is about 1.2 w/m c.

** θ_{CA} values shown in this table are typical values. The actual θ_{CA} values depend on the air flow in the system (which is typically unsteady, non-uniform and turbulent) and thermal interactions between Pentium CPU and surrounding components through PCB and the ambient.

5.0 DESIGNING FOR THERMAL PERFORMANCE

At this point the application note turns from describing the characteristics that define thermal performance to describing how designers should use these characteristics to assess thermal requirements of PC system designs. The Pentium processor specifies a maximum case temperature, T_C , of 70°C @ 66 MHz. This case temperature limit along with the Pentium processor's power and thermal resistance characteristics can be used to determine the ambient temperature required to keep the Pentium processor operating within its specified limits. Using these parameters in the following equations:

$$T_A = T_C - (P \cdot \theta_{CA})$$

$$T_A = 70^\circ\text{C} - (16\text{W} \cdot 10.5^\circ\text{C/W})$$

$$T_A = -98^\circ\text{C}$$

The maximum ambient temperature required in a Pentium processor system without any additional thermal enhancement is -98°C at 66 MHz. Obviously, this ambient temperature is impractical and unachievable in a PC system. In order to be able to maintain the case temperature at 70°C in a typical system ambient with air temperature of 40°C to 45°C, the thermal resistance between the case and the ambient must be reduced.

3

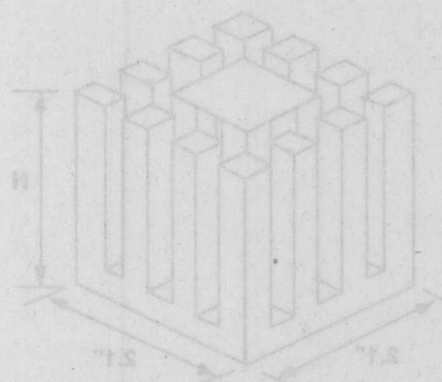


Figure 4. Recommended Combinations

5.1 Heat Sinks

The most common way to improve the package thermal performance is to increase the surface area of the device by attaching a large piece of metal (a heat sink) to the package. The heat sink is usually made of Aluminum and is chosen for its price/thermal-performance ratio. There are materials that offer higher conductivity such as copper, but cost becomes prohibitive. To maximize the flow of heat for a given junction temperature rise over the ambient temperature, the thermal resistance from heat sink to air can be reduced by a) maximizing the surface area, and b) maximizing the air flow across the surface area (maximizing air flow through heat sink fins in most cases).

Intel has used test data to determine what size of heat sink and airflow is needed to properly cool a Pentium processor system. The data was derived assuming an

adhesive attach process that offers thermal resistance of about $0.2^{\circ}\text{C}/\text{W}$.

The testing was done in a wind tunnel in the configuration (in Figure 3) where the heat sink was mounted on a real Pentium processor package with a thermal die mounted inside to generate the 16W of power. The package is then mounted in a socket which is soldered to a 2-layer PCB that brings power to the die.

Based on these tests, three specific heat sink and airflow combinations have been identified that properly dissipate the Pentium processor's 16W and maintains a case temperature below 70°C @ 66 MHz. The three heat sinks are shown in Figure 4.

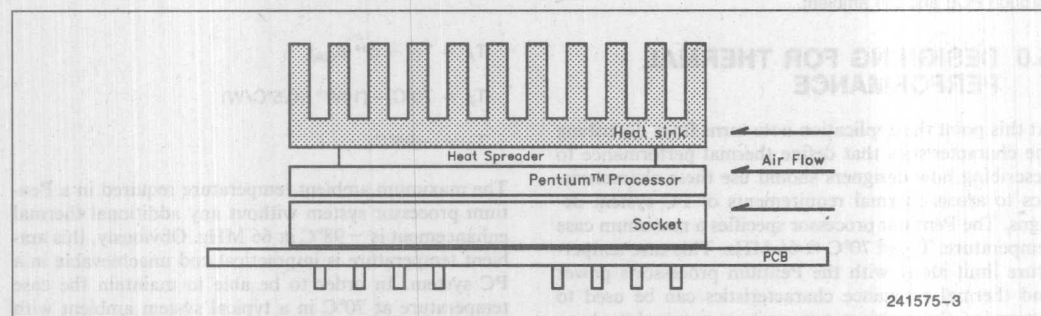


Figure 3. Improving Thermal Performance

Heat sink A: $H = 1.4"$ for 200 LFMs of Air
 Heat sink B: $H = 1.05"$ for 300 LFMs of Air
 Heat sink C: $H = 0.85"$ for 400 LFMs of Air
 Assumption: Air Temp = 45°C

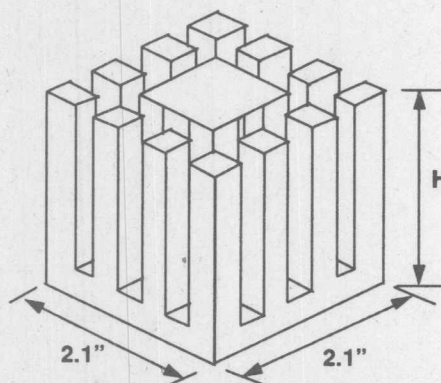


Figure 4. Recommended Combinations

In addition, testing has been done to provide more general guidelines which allow deviating from the above conditions. These guidelines allow systems to derive various combinations of heat sink size and airflow that ensure the Pentium processor thermal specifications are met. For example, by increasing the heat sink x-y dimensions and extending it over the package footprint, the heat sink height can be reduced while maintaining the same thermal performance as the taller heat sink with the same footprint as that of the package. The first three charts (Figures 5, 6, 7) show the thermal resistance as a function of heat sink size and airflow. The last three charts (Figures 8, 9, 10) show the power dissipation

achievable with a given heat sink size and airflow. The power dissipation calculations assume $T_C = 70^\circ\text{C}$ @ 66 MHz, $T_A = 45^\circ\text{C}$, and $\theta_{JC} = 0.6^\circ\text{C/W}$.

$$P_{\max} = (T_C - T_A) / \theta_{CA} = 25 / \theta_{CA}$$

A key assumption in all of these calculations is that a perfect thermal connection can be achieved between the case and the heat sink. One can extrapolate the heat sink solutions by adding the additional thermal resistance of any chosen heat sink attach process. See Appendix B for case to heat sink thermal interface options.

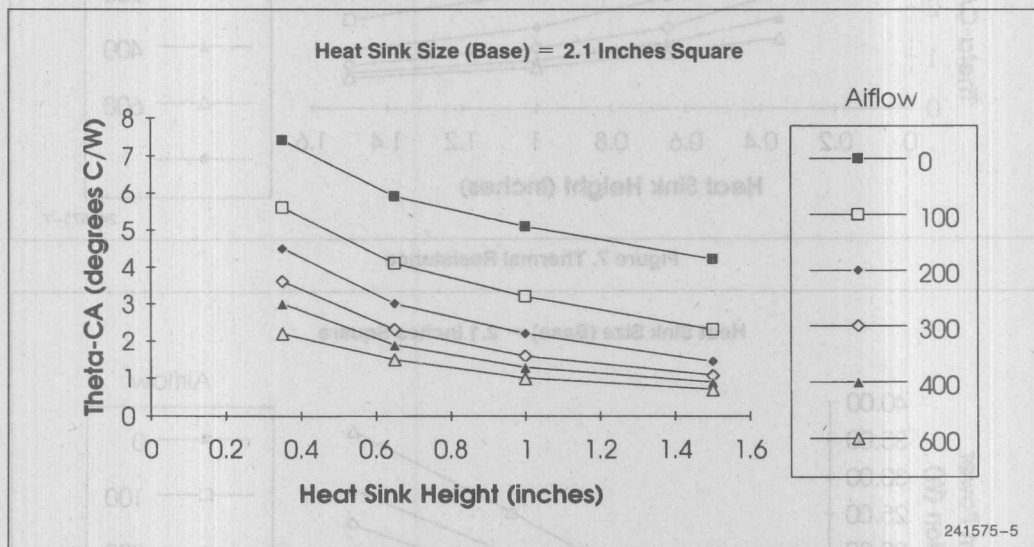


Figure 5. Thermal Resistance

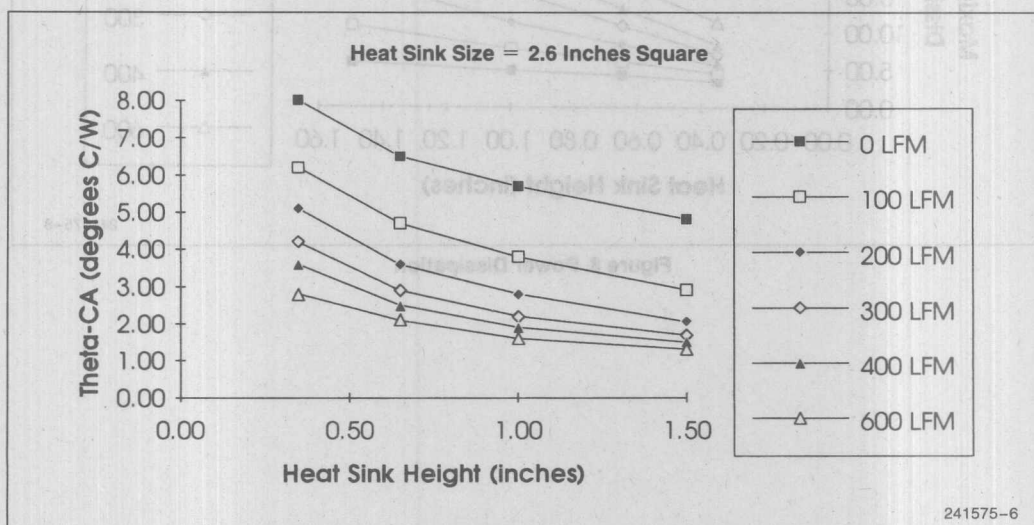


Figure 6. Thermal Resistance

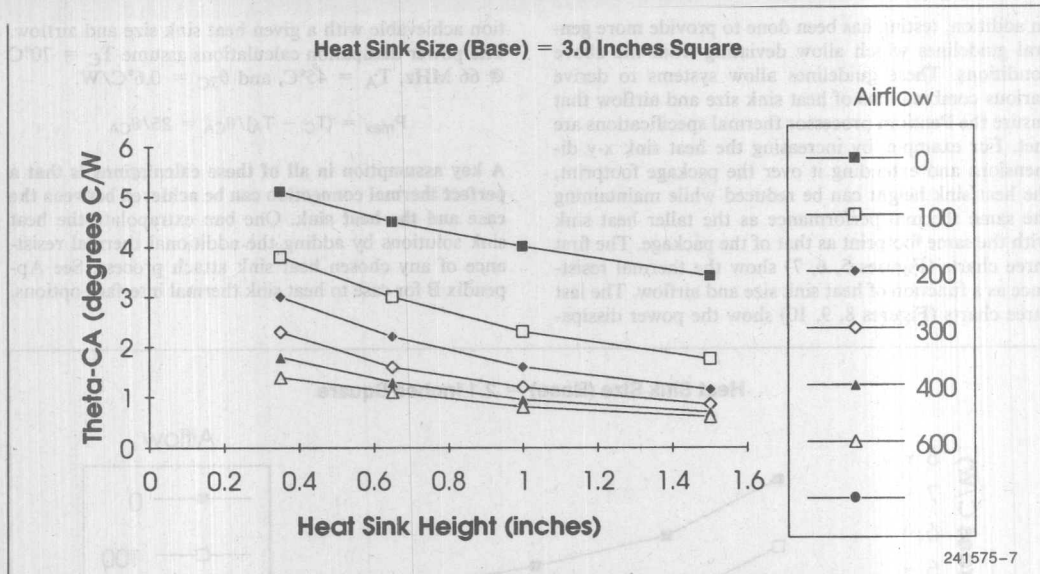


Figure 7. Thermal Resistance

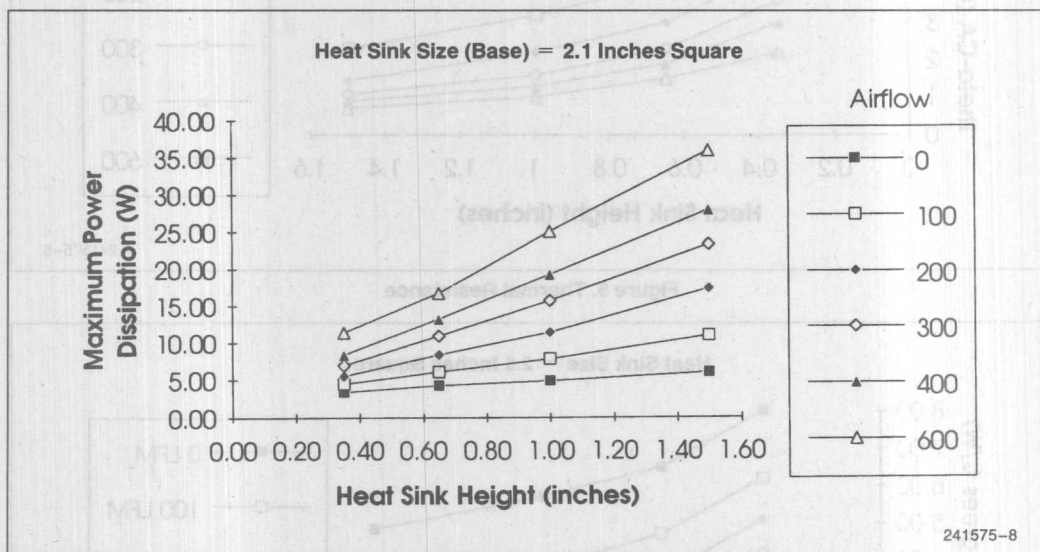


Figure 8. Power Dissipation

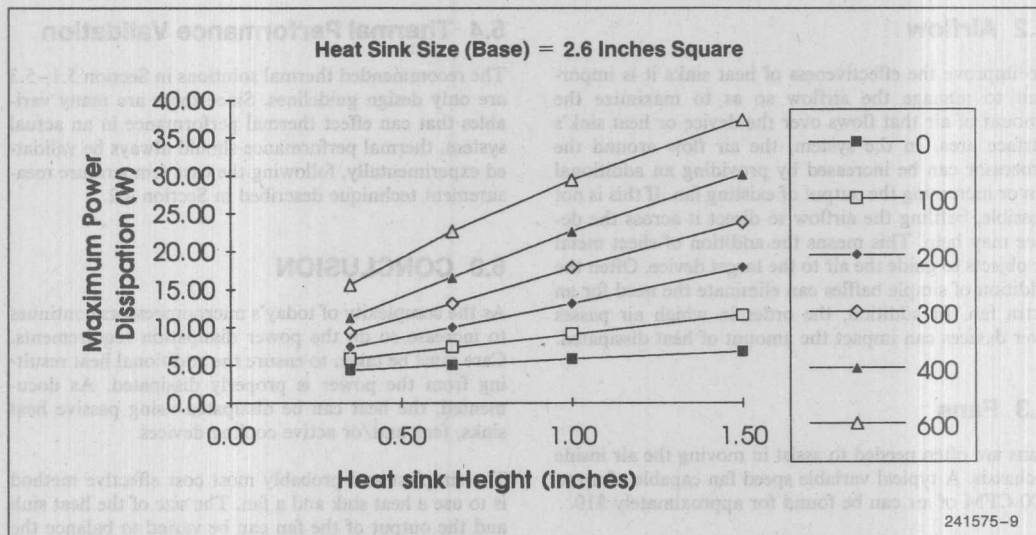


Figure 9. Power Dissipation

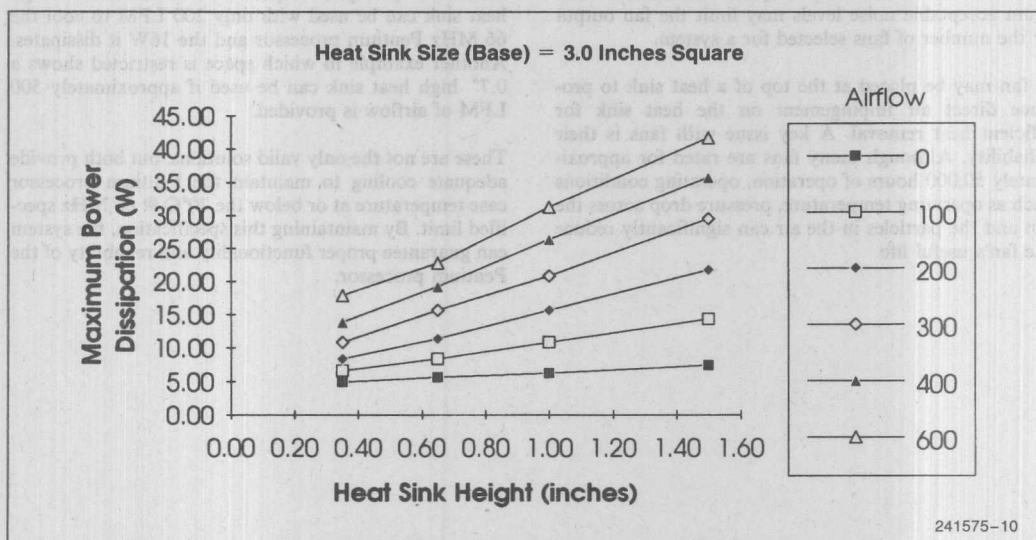


Figure 10. Power Dissipation

5.2 Airflow

To improve the effectiveness of heat sinks it is important to manage the airflow so as to maximize the amount of air that flows over the device or heat sink's surface area. In the system, the air flow around the processor can be increased by providing an additional fan or increasing the output of existing fan. If this is not possible, baffling the airflow to direct it across the device may help. This means the addition of sheet metal or objects to guide the air to the target device. Often the addition of simple baffles can eliminate the need for an extra fan. In addition, the order in which air passes over devices can impact the amount of heat dissipated.

5.3 Fans

Fans are often needed to assist in moving the air inside a chassis. A typical variable speed fan capable of up to 100 CFM of air can be found for approximately \$10.

The airflow rate is usually directly related to the acoustic noise level of the fan and system. Therefore maximum acceptable noise levels may limit the fan output or the number of fans selected for a system.

A fan may be placed at the top of a heat sink to produce direct air impingement on the heat sink for efficient heat removal. A key issue with fans is their reliability. Although many fans are rated for approximately 50,000 hours of operation, operating conditions such as operating temperature, pressure drop across the fan and the particles in the air can significantly reduce the fan's useful life.

5.4 Thermal Performance Validation

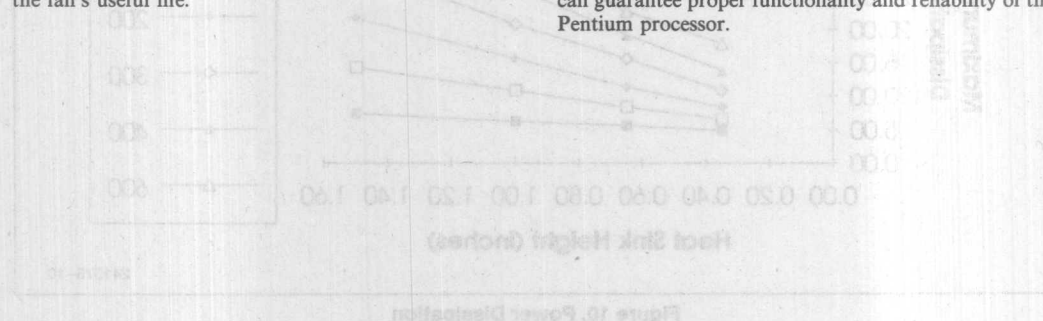
The recommended thermal solutions in Section 5.1–5.3 are only design guidelines. Since there are many variables that can effect thermal performance in an actual system, thermal performance should always be validated experimentally, following the case temperature measurement technique described in Section 4.2.

6.0 CONCLUSION

As the complexity of today's microprocessors continues to increase so do the power dissipation requirements. Care must be taken to ensure the additional heat resulting from the power is properly dissipated. As documented, the heat can be dissipated using passive heat sinks, fans and/or active cooling devices.

The simplest and probably most cost effective method is to use a heat sink and a fan. The size of the heat sink and the output of the fan can be varied to balance the tradeoffs between size and space constraints versus noise. For example, if space is available a 1.4" high heat sink can be used with only 200 LFM to cool the 66 MHz Pentium processor and the 16W it dissipates. Another example in which space is restricted shows a 0.7" high heat sink can be used if approximately 500 LFM of airflow is provided.

These are not the only valid solutions, but both provide adequate cooling to maintain the Pentium processor case temperature at or below the 70°C @ 66 MHz specified limit. By maintaining this specification, the system can guarantee proper functionality and reliability of the Pentium processor.



APPENDIX A EXAMPLES

Thermal management examples, designing with the Pentium processor. Using the best known methods.

Appendix Goal

The goal of this appendix is to measure the operating temperatures in a real system versus the wind tunnel laboratory measurements. These experiments are done with heat sinks that are similar to the ones suggested in Section 5.1 of the main document. The thermocouples and attachment methods suggested in Section 4.2 of the main document are also used. The appendix begins by reviewing the variables that the system designer has control over and uses tables to describe thermal resistance in the context of where the system designer can have the most effect. The importance of the case to heat sink thermal interface and correct attachment methods are reviewed and different options given. The appendix proceeds to describe the system used for these tests and the tools and equipment needed. The lab set up procedures are discussed in detail and the measurements are presented with comments at the conclusion.

3

WHAT ARE THE VARIABLES?

Table A-1 shows the cooling options that customers can control when designing a system. From Table A-1 it is obvious that changing the heat sink and air flow are the two most effective ways for a system designer to affect the thermal performance of a system.

Table A-1. Variables

COOLING OPTIONS UNDER CUSTOMER CONTROL	
Variables	Options for Cooling
Device	<ul style="list-style-type: none">• Use Power Management SW in the System• Clock the Device at a Lower Speed
Heat Sink	<ul style="list-style-type: none">• Increase Heat Sink Area, Width or Height• Use Interface Materials with Lower Thermal Resistance• Use Active Cooling Devices• Use a Combination of Active and Passive Cooling Devices
Air Flow	<ul style="list-style-type: none">• Increase Air Flow• Manage Air Flow• Place Hottest IC's near Highest Airflow
Ambient Temperature	<ul style="list-style-type: none">• Place System in a Controlled Climate

Figure A-1 sums up the thermal equation picture succinctly. Looking at Figure A-1 reiterates the previous statement that increasing the heat sink size and air flow rate provide the largest thermal performance improvements. In addition it shows the variables that are constant. Note that the θ_{JC} (junction-to-case thermal resistance) of the Pentium processor is fixed and a system designer can have no effect on this parameter. Also note that the θ_{CS} (case-to-heat sink thermal resistance) is a constant. Even though θ_{CS} is shown as a constant in Figure A-1 it can move up and down the Y axis depending on the interface material chosen. The case to heat sink interface is critical to the overall success of the thermal solution and cannot be overlooked. The next section will go into detail on this subject.

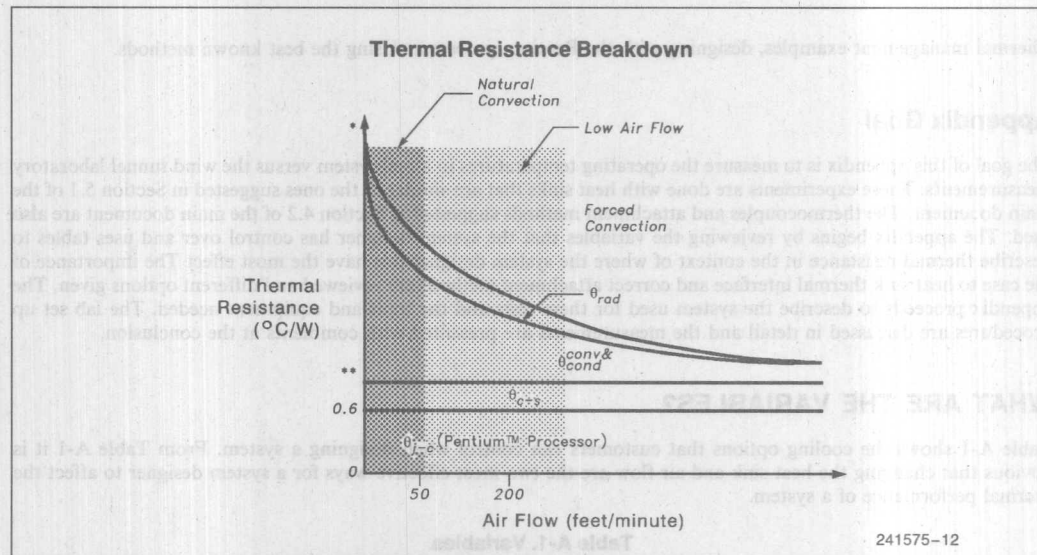


Figure A-1. Thermal Resistance

Options for Cooling	Variable
<ul style="list-style-type: none"> • Use Power Management SW in the System • Clock the Device at a Lower Speed 	Device
<ul style="list-style-type: none"> • Increase Heat Sink Area, Width or Height • Use Interface Materials with Lower Thermal Resistance • Use Active Cooling Devices • Use a Combination of Active and Passive Cooling Devices 	Heat Sink
<ul style="list-style-type: none"> • Increase Air Flow • Manage Air Flow • Place Hottest IC's near Highest Airflow 	Air Flow
<ul style="list-style-type: none"> • Place System in a Controlled Climate 	Ambient Temperature

The main of purpose Figure A-2 is to show that packages and heat sinks are not perfectly flat and this requires that the air gap be filled with an interface material that has a lower thermal resistance than air. The whole point is to try and minimize the contact thermal resistance. The different types of thermal interface materials are listed to show the wide array of materials available to the system designer. Intel's data books have a mechanical section that list the flatness of the package. Heat sink vendors should be able to provide specifications for their heat sink offerings.

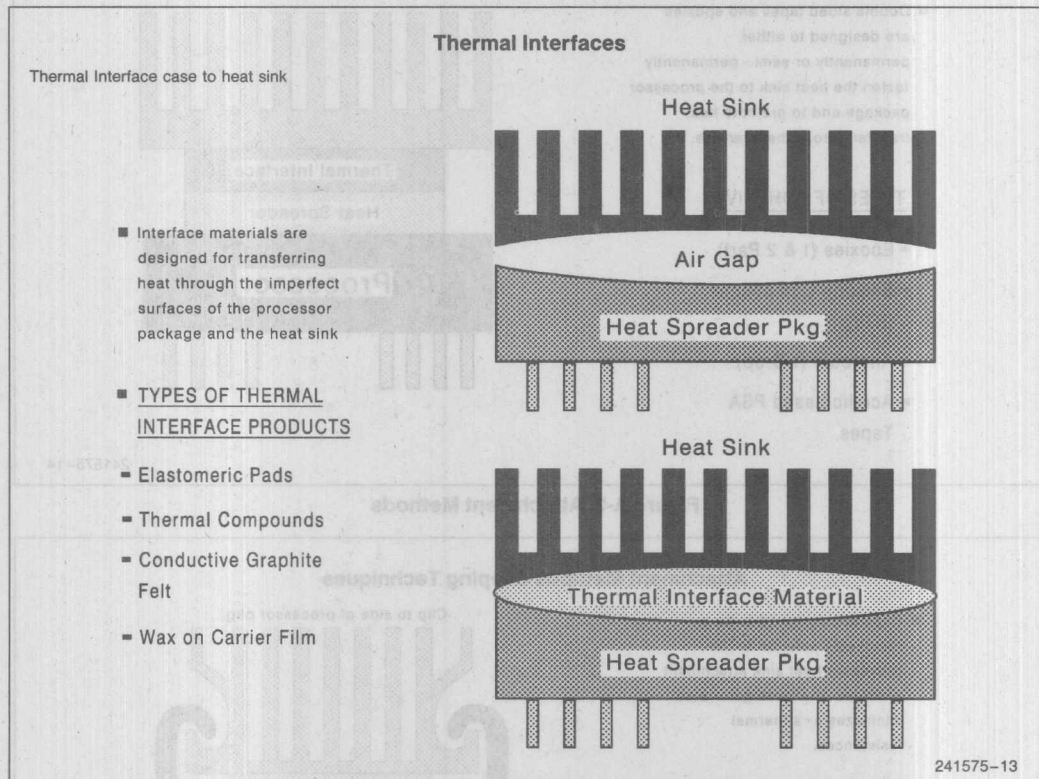


Figure A-2. Thermal Interfaces



Figure A-4. Attachment Methods

The next section (Figures A-3 through A-5) covers attachment methods which generally fall into the categories shown; epoxies, double sided tapes or manual clips to either chip or socket.

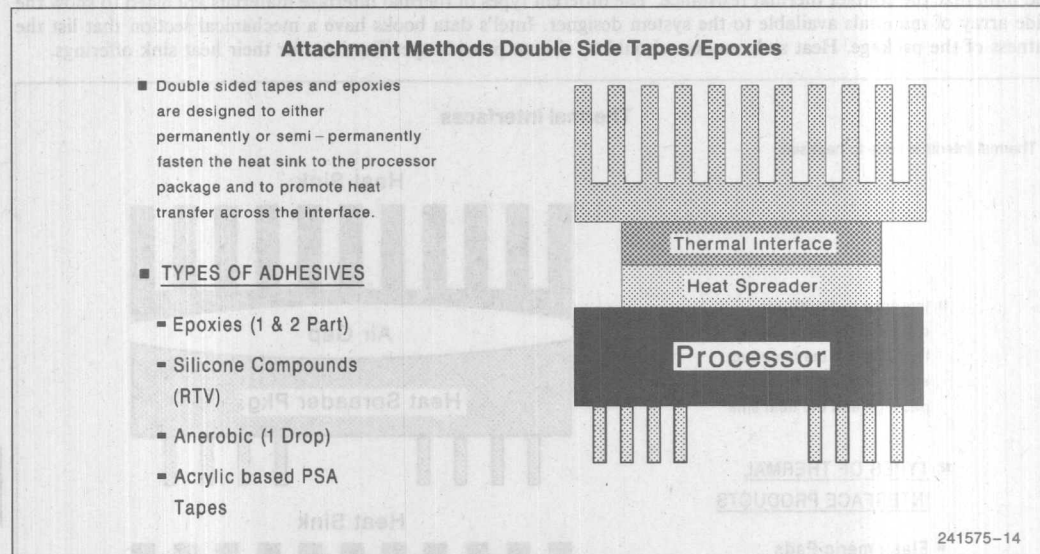


Figure A-3. Attachment Methods

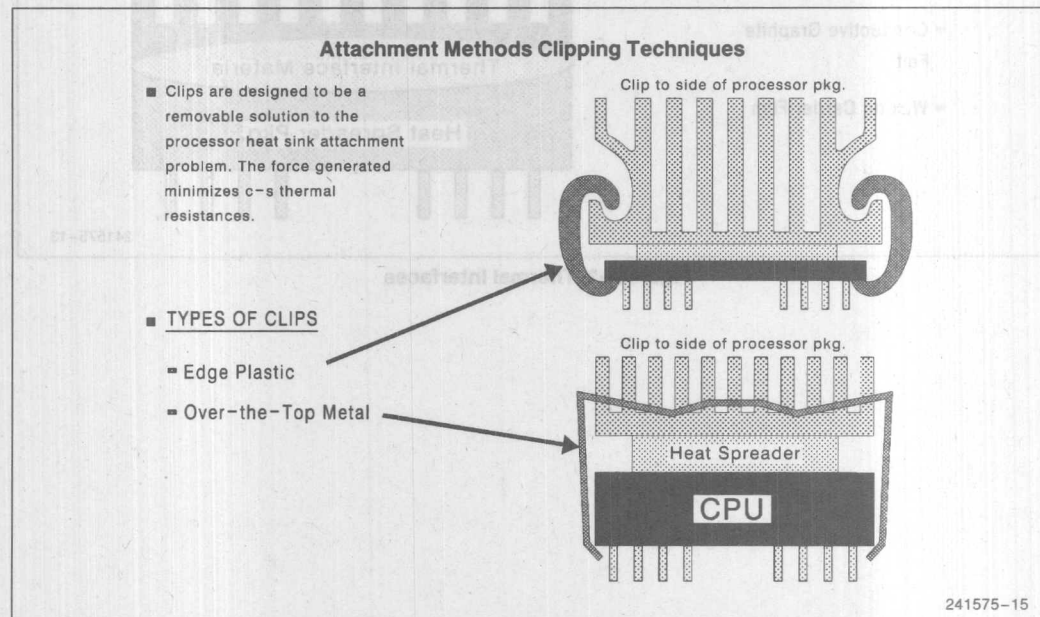


Figure A-4. Attachment Methods

Note that some clips don't allow the package to be pushed all the way into the socket and this could be a problem with short lead packages. The main advantage of this type of system is that a low profile socket can be used to lower the height of the processor heat sink assembly.

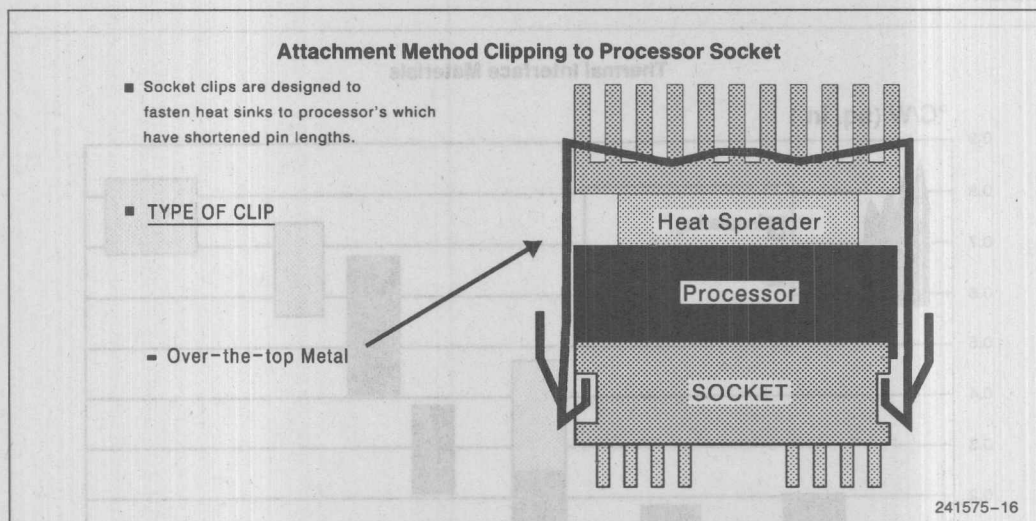


Figure A-5. Attachment Methods

Table A-2 lists pros and cons of the different attachment methods covered.

Table A-2. Attachment Methods

ATTACHMENT METHODS		
	ADVANTAGES	DISADVANTAGES
DOUBLE SIDED TAPES	<ul style="list-style-type: none"> • Quick to Use • Low Installed Cost • Compliant 	<ul style="list-style-type: none"> • High Thermal Resistance • Requires Flat Interfaces • Assembly Contact Pressure
EPOXIES	<ul style="list-style-type: none"> • Low Potential Thermal Resistance • Low Contact Pressure 	<ul style="list-style-type: none"> • Mixing, Curing, Messy • Timing Consuming (if not automated) • CTE Stress, High Rigidity • Variable Thickness (theta)
CLIPS	<ul style="list-style-type: none"> • Centralized Pressure Points • Removable • Easily Installed • Solution to Upgrade • Accommodates Wide Tolerance 	<ul style="list-style-type: none"> • Removable • Force Limits vs Assembly • Insufficient Shock and Vibration Data • Potential for Loss of Pressure

though thermal grease has a deserved reputation for being messy and harder to control it still performs well as a thermal interface. All the examples that are shown in Appendix A use thermal grease as the case to heat sink interface.

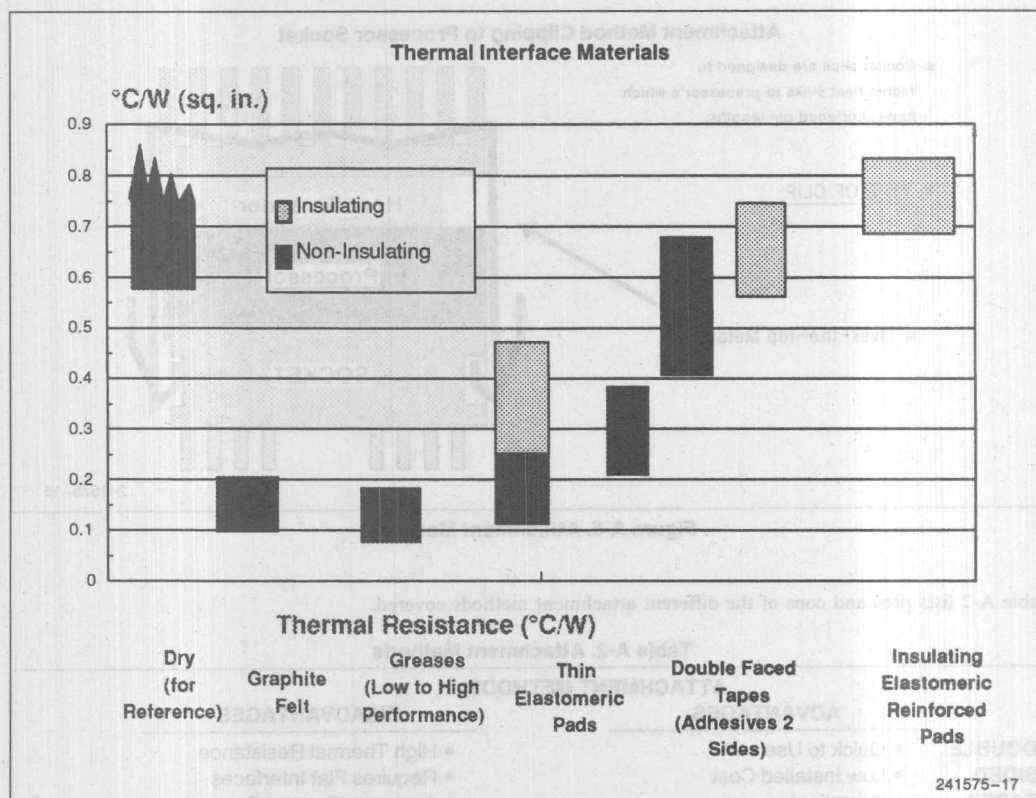


Figure A-6. Materials

The next step is to choose a heat sink. Figure A-7 shows the wide range of choices and the cost associated with each technology.

Now that all the variables and options are known for this problem we can proceed on to do some real system measurements using the recommendations and data shown in the first part of this application note.

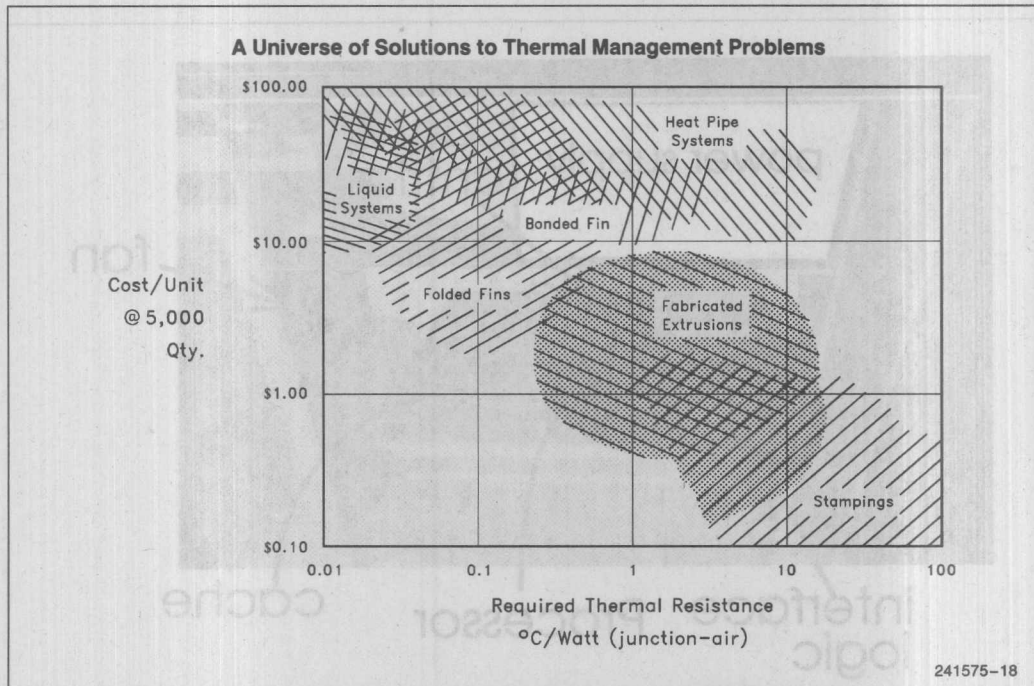


Figure A-7. Solutions

Examples

For all the examples in this section we used a 40 MHz system with a Pentium processor and 256K cache. A picture of the system under test is shown in Figure A-8 with the covers off to show the placement of the Pentium processor and the associated cache components. A 40 MHz system was used because it was the only one available at the time the testing was done.

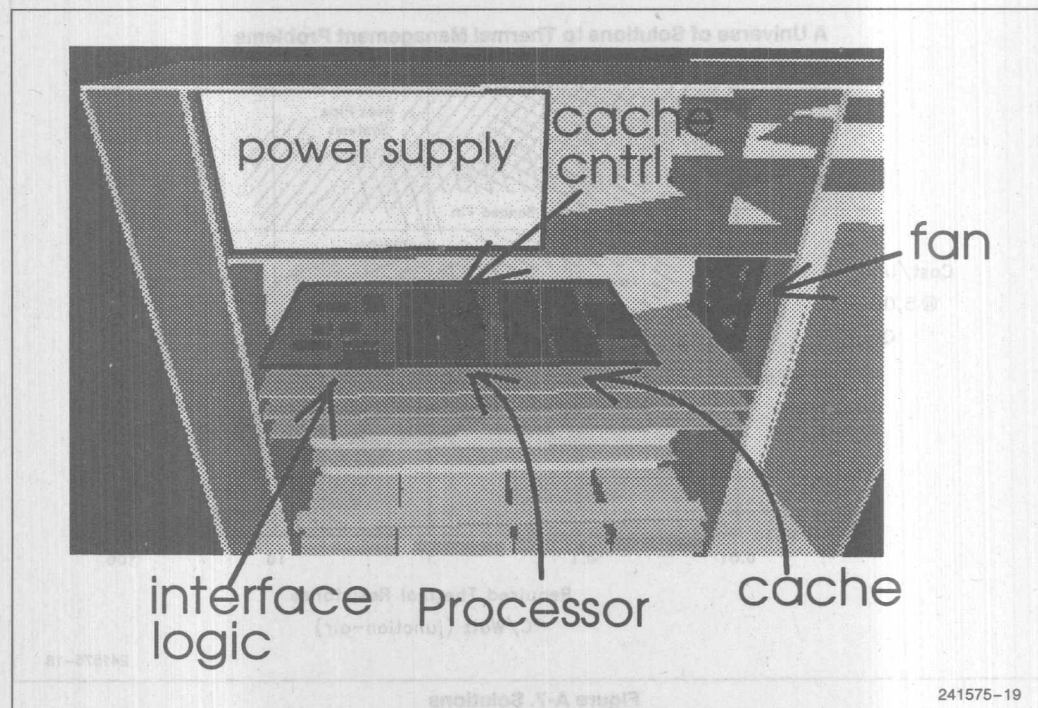


Figure A-8. Pentium™ Processor System

Objectives

- To measure a Pentium processor system operating under real working conditions.
- To compare the measured results to the predicted results shown in the beginning of this application note. The reader should always keep the main goal in mind; **the main goal is always to meet the case temperature specification for the Pentium processor.** Any combination of heat sink and air flow rate is fine as long as the case temperature specification is met. The heat sinks used in test #1 thru #4 will match the suggested heat sinks as close as possible to accurately correlate with the wind tunnel data. This is meant to illustrate how a system designer might start by using the suggested heat sinks and air flow rates as starting points to thermally tune their particular system. Test #5 uses a heat sink and a fan combination. The fan heat sink is best described as a fan attached directly to the heat sink on the Pentium processor. It is an active device used for spot cooling ICs. We will concentrate on traditional passive heat sink solutions with only one set of measurements being done for a fan heat sink assembly.

Tools and Equipment

1. Pentium processor-based system running at 40 MHz.
2. Hot wire anemometer to measure airflow rate.
3. Thermocouples and high thermal conductivity cement as recommended in the application note.
4. Homemade jig for accurate and repeatable attachment of the thermocouples to the package.
5. Homemade power supply isolation socket for setting the V_{CC} and reading the I_{CC} of the processor independently of the rest of the system.
6. Adjustable power supply with adequate current capabilities and both current and voltage read out.
7. Multimeter to read the voltage and current.
8. Cables to connect everything up.
9. Software test suite that simulates "worst case conditions for a typical real application." In this case it was Microsoft Excel and Word for Windows test suites.
10. Drill and drill bits.
11. Thermal grease.

The lab procedure was as follows:

Preparing the System

1. Load the test software on the system disk (or floppy) and make sure everything runs correctly before you start. After everything works satisfactorily proceed to the next step.
2. Remove the covers, choose several places (random) around the processor to measure the air flow of the system. Then drill holes large enough to allow the

anemometer to be inserted. Five holes were drilled in the system cover.

3. In this case we had a 12" long $\frac{1}{4}$ " diameter directional anemometer. To get more repeatable measurements the shaft of the probe was marked with a pencil to get the same depth, into the box, for each measurement
4. We then removed the processor card from the chassis (use anti-static procedures to prevent IC damage).
5. Remove the Pentium processor from the card and install the isolation socket.

Preparing the Pentium Processor for Testing

1. Using the jig carefully attach the thermocouple to the center of the processor package using cement and let it cure as recommended by the manufacturer of the cement.
2. Drill holes no larger than 0.125" in the centers of the heat sinks to be tested just large enough to get the thermocouple wires through the hole. In the case of the fan heat sink, the fan was removed and the heat sink was drilled the same as the others and then re-assembled. Each of the holes were counter sunk on the bottom to better conform to the tear drop shape the thermocouple and cement naturally forms into. The idea is to not disturb or break the contact between the cement and the package. If it is broken or cracked the measurements will be incorrect
3. Apply the thermal grease (less than 0.004" thick) evenly, with no voids, to the processor package.
4. Slide the heat sink down the thermocouple wires being careful not to disturb the thermocouple while at the same time firmly seating the heat sink to the package. Attach the plug for the temperature meter to the other end of the thermocouple wire terminals.
5. Re-install the processor/thermocouple/heat sink assembly into the isolation socket on the processor board, again being careful not to disturb the thermocouple connection.

Preparing for Measurements

1. Re-install the processor card into the system.
2. Connect the power supply wires to the power supply and the isolation socket.
3. Connect the multimeter to the the power supply to monitor the V_{CC} and set the power supply meter to measure I_{CC} .
4. Connect the thermocouple to the meter.
5. Turn on the processor power supply and then the system supply.
6. Wait for the system to boot and then run the test software.

Thermal Measurements

The next step was to determine the baseline airflow in the system without a heat sink attached to the processor. Measure the airflow at several locations using the access holes in the system and the marks on the probe to ensure accurate placement of the probe and repeatability of the measurements. Table A-3 shows the results. Be cautious when placing the fan in a system relative to the processor. All fans have a dead spot (low airflow) in the center of the fan. Avoid the dead spot. Even several inches away from the fan the dead spot can influence airflow considerably.

Test #1

The next step is to compare how close the suggested values and tables are to the measured results. Use the formulas described in the beginning of the application note and the values from Table A-4.

$$P_D = V_{CC} \cdot I_{CC} = (1.82 \cdot 4.89) = 8.827W$$

$$\theta_{JC} = (T_J - T_C) / P_D = 0.6$$

$$\theta_{JA} = (T_J - T_A) / P_D$$

$$\theta_{CA} = \theta_{JA} - \theta_{JC} = [(T_J - T_A) - (T_J - T_C)] / P_D = [T_C - T_A] / P_D$$

$$\theta_{CA} = (55.3 - 29) / (1.8 \cdot 4.85) = 24 / 8.827 = 2.97$$

$\theta_{CA} = 2.7^\circ C/W$ is the measured value in the system for this configuration.

Table A-3. Baseline Airflow

Airflow Measured, LFM	120-160
Location of probe	~ 2 inches (upstream from the fan) @ center of the processor (above the heat sink)

Table A-4. Test Conditions Test #1

Heat Sink Size, Inches H x W	Temperature, degrees C			I _{CC} Amps	V _{CC} Volts	Air Flow LFM
	Room T _A	System T _A	Case T _C			
1.2 x 2.1 sq.	23	29	55.3	1.82	4.85	100-150

The graph (Figure A-9) from the application note, for heat sink size size of 2.1" x 2.1", is used to compare the predicted θ_{CA} , for a 1.2" tall heat sink, to the measured value of θ_{CA} .

The predictions from the graph (Figure A-9) are:

$$\theta_{CA} = 2.9^{\circ}\text{C/W @ 100 LFM}$$

$$\theta_{CA} = 2.4^{\circ}\text{C/W @ 150 LFM}$$

$$\theta_{CA} = 1.9^{\circ}\text{C/W @ 200 LFM}$$

And the measured value is:

$$\theta_{CA} = 2.97^{\circ}\text{C/W @ 100-150 LFM}$$

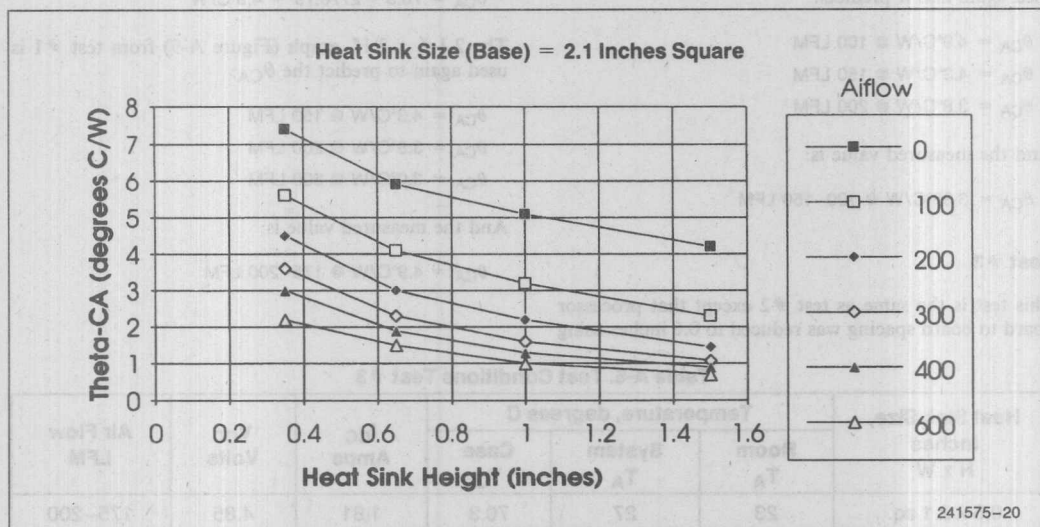


Table A-5. Test Conditions Test #2

Heat Sink Size, Inches H x W	Temperature, degrees C			I _{CC} Amps	V _{CC} Volts	Air Flow LFM
	Room T _A	System T _A	Case T _C			
0.5 x 2.1 sq.	22	29	58.3	1.81	4.85	100-150

Test #2

The same test only the heat sink height was reduced to 0.5 inch height.

$$\theta_{CA} = (T_C - T_A)/P_D$$

$$\theta_{CA} = 58.3 - 29/8.79 = 3.33^\circ\text{C/W}$$

The 2.1" x 2.1" graph (Figure A-9) from test #1 is used again and it predicts:

$$\theta_{CA} = 4.9^\circ\text{C/W @ 100 LFM}$$

$$\theta_{CA} = 4.3^\circ\text{C/W @ 150 LFM}$$

$$\theta_{CA} = 3.8^\circ\text{C/W @ 200 LFM}$$

And the measured value is:

$$\theta_{CA} = 3.33^\circ\text{C/W @ 100-150 LFM}$$

Test #3

This test is the same as test #2 except that processor board to board spacing was reduced to 0.6 inches using

a cardboard baffle to simulate a system with very tight board spacing. An existing system that is upgrading from an Intel 486 processor to the Pentium processor might have this type of spacing. Note that this particular configuration actually has more airflow than test #2. It could have just as easily been lower. It all depends on the particular system being measured.

$$\theta_{CA} = (T_C - T_A)/P_D$$

$$\theta_{CA} = 70.3 - 27/8.79 = 4.9^\circ\text{C/W}$$

The 2.1" x 2.1" graph (Figure A-9) from test #1 is used again to predict the θ_{CA} :

$$\theta_{CA} = 4.3^\circ\text{C/W @ 150 LFM}$$

$$\theta_{CA} = 3.8^\circ\text{C/W @ 200 LFM}$$

$$\theta_{CA} = 3.0^\circ\text{C/W @ 300 LFM}$$

And the measured value is:

$$\theta_{CA} = 4.9^\circ\text{C/W @ 175-200 LFM}$$

Table A-6. Test Conditions Test #3

Heat Sink Size, Inches H x W	Temperature, degrees C			I _{CC} Amps	V _{CC} Volts	Air Flow LFM
	Room T _A	System T _A	Case T _C			
0.5 x 2.1 sq.	23	27	70.3	1.81	4.85	175-200

Table A-7. Test Conditions Test #4

Heat Sink Size, Inches H x W	Temperature, degrees C			I _{CC} Amps	V _{CC} Volts	Air Flow LFM
	Room T _A	System T _A	Case T _C			
0.65 x 3.1 sq.	23	29	55.3	1.8	4.85	100-140

Test #4

This test uses a 0.65" tall heat sink that is 3.1" sq. This type of heat sink might be used when height is limited and there is room to spread out by adding more area to the heat sink base.

$$\theta_{CA} = (T_C - T_A) / P_D$$

$$\theta_{CA} = (55.3 - 29) / 8.73 = 3.0$$

The 3.0" x 3.0" graph (Figure A-10) from the application note is used since it is similar to the heat sink used. The 3.0" x 3.0" graph predicts:

$$\theta_{CA} = 3.0^{\circ}\text{C/W} @ 100 \text{ LFM}$$

$$\theta_{CA} = 2.6^{\circ}\text{C/W} @ 150 \text{ LFM}$$

And the measured value is:

$$\theta_{CA} = 3.0^{\circ}\text{C/W} @ 100\text{--}140 \text{ LFM}$$

Test #5

The last test was done using a fan/heat sink assembly that has become popular for prototyping, debug and spot cooling in some situations. We were not able to measure the airflow on the processor with this configuration because the air flow is not directional enough to get a reading with the probe available. The case temperature however was monitored by mounting a thermocouple in the same manner used above. We did modify the setup by bringing the thermocouple wires out the side to clear the fan. This will change the measurements the thermocouple produces and should be factored into any data. We do not have any wind tunnel data on the fan/heat sink combination. Note that the case temperature is within specification.

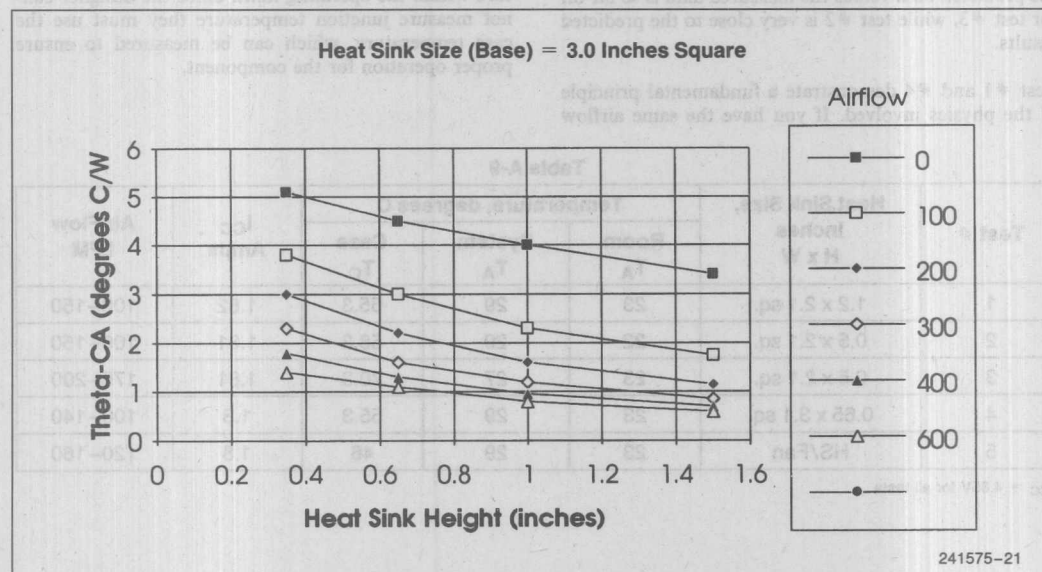


Figure A-10. Thermal Resistance

Table A-8. Test Conditions Test #5

Heat Sink Size, Inches H x W	Temperature, degrees C			I _{CC} Amps	V _{CC} Volts	Air Flow LFM
	Room T _A	System T _A	Case T _C			
HS/Fan	23	29	46	1.8	4.85	120-160

Conclusion

Table A-9 shows all the tests in one table. The data shows that the suggestions in the application note are a very good starting point to begin tuning any Pentium processor system and that there is no one cookbook answer that fits all systems because of the complexity of air flow and variations from each type of system. Indeed the results show that airflow can be changed dramatically even in the same system by changing one variable. For example test #2 and #3 are exactly the same except that board to board spacing was reduced significantly. Note that case temperature rose significantly even though the airflow sensor was reading a higher value. This suggests that the airflow through the heat sink was lower even though the anemometer, 2 inches away, was reading higher airflow at its position. Note also that test #2 more closely approximates the wind tunnel test setup because it has open space above the board instead of a board nearby. This is also why the predicted data versus the measured data is so far off for test #3, while test #2 is very close to the predicted results.

Test #1 and #4 demonstrate a fundamental principle of the physics involved. If you have the same airflow

and must reduce the height of the heat sink, you have to spread out the area of the heat sink to compensate for the reduced height. Test #1 uses a 1.2" height heat sink that is the same size as the package. Test #4 was able to produce the same case temperature with a shorter heat sink and more area.

Test #5 demonstrates that a fan/heat sink assembly can spot cool effectively if you have enough space above and around it to allow the required back pressure. This is the only active device tested. If you look back at the "A Universe of Solutions to Thermal Management Problems" (Figure A-7) chart you will see the reason why. While the Pentium processor is at the outer envelope of passive cooling, this method of cooling still offers lower cost, power usage and reliability in most cases.

Most of all the system designer should never lose sight of the real goal which is to keep the junction temperature within the operating limit. Since the designer cannot measure junction temperature they must use the case temperature, which can be measured to ensure proper operation for the component.

Table A-9

Test #	Heat Sink Size, Inches H x W	Temperature, degrees C			I _{cc} Amps	Air Flow LFM
		Room T _A	System T _A	Case T _C		
1	1.2 x 2.1 sq.	23	29	55.3	1.82	100-150
2	0.5 x 2.1 sq.	22	29	58.3	1.81	100-150
3	0.5 x 2.1 sq.	23	27	70.3	1.81	175-200
4	0.65 x 3.1 sq.	23	29	55.3	1.8	100-140
5	HS/Fan	23	29	46	1.8	120-160

V_{CC} = 4.85V for all tests.

Test #	Heat Sink Size, Inches H x W	Room T _A	System T _A	Case T _C	I _{cc} Amps	V _{CC} Volts	Air Flow LFM
1	1.2 x 2.1 sq.	23	29	55.3	1.82	4.85	100-150

APPENDIX B HEAT SINK VENDORS

Aavid Engineering

One Kool Path
P.O. Box 400
Laconia, NH 03247
(603) 528-3400
(603) 525-1478 (FAX)

Contact: Gary F. Kuzmin (Product Marketing Manager)

EG&G Wakefield Engineering

60 Audubon Rd.
Wakefield, MA 01880
(617) 245-5900
(617) 246-0874 (FAX)

Contact: David Saums (Marketing Manager)

IERC

135 W. Magnolia Blvd.
Burbank, CA 91502
(818) 842-7277
(818) 848-8872 (FAX)

Contact: Guy R. Addis (Western Region Applications Engineer)

Thermalloy

2021 W. Valley View Lane
Dallas, TX 75234-8993
(214) 243-4321

Contact: Larry Tucker (VP of Sales and Marketing)



AP-481

APPLICATION NOTE

Designing with the Pentium™ Processor, 82496 Cache Controller and 82491 Cache SRAM CPU-Cache Chip Set

JIM REILLY
TECHNICAL MARKETING

November 1993

Designing with the Pentium™ Processor, 82496 Cache Controller and 82491 Cache SRAM CPU-Cache Chip Set

CONTENTS

	PAGE
1.0 INTRODUCTION	3-199
2.0 A/C SPECIFICATIONS OF THE CHIP SET	3-199
2.1 Optimized Interface	3-199
2.1.1 Flight Time Specification	3-199
2.1.1.1 Purpose of Flight Time Specification	3-199
2.1.1.2 Definition of Flight Time	3-199
2.1.1.3 Clock Skew	3-202
2.1.2 Signal Quality Specifications	3-202
2.1.2.1 Overshoot	3-203
2.1.2.2 Time Beyond Supply ...	3-203
2.1.2.3 Ringback	3-204
2.1.2.4 Settling Time	3-204
2.1.2.5 Group Averages	3-204
2.2 External Interface	3-204
2.2.1 Output Valid Delay and Float Time	3-204
2.2.2 Input Setup and Hold Time ..	3-205
3.0 I/O BUFFER MODELS	3-206
3.1 Description of the First Order I/O Buffer Model	3-206
4.0 HIGH FREQUENCY DESIGN CONSIDERATIONS	3-207
4.1 Printed Circuit Board	3-210
4.2 Transmission Line Behavior	3-212
4.2.1 Signal Propagation and Reflection	3-212
4.2.2 Crosstalk	3-216
5.0 CHIP SET DESIGN	3-218
5.1 Simulation Environment	3-219
5.1.1 Simulation Requirements ...	3-219
5.2 Routing Signal Traces for Their Optimal Performance	3-220
5.2.1 Rules for Optimizing Signal Routing	3-221

CONTENTS

	PAGE
5.2.2 Determining the Optimal Net	3-221
5.2.3 Serpentine Structures	3-225
6.0 EXAMPLE: DESIGNING THE A12 NET FOR THE CPU-CACHE CHIP SET	3-227
7.0 256K CPU-CACHE CHIP SET OPTIMIZED INTERFACE LAYOUT DESIGN EXAMPLE	3-234
7.1 Layout Objectives	3-235
7.2 Component Placement	3-235
7.3 Signal Routing/Topologies	3-236
7.4 Board/Trace Properties	3-255
7.5 Design Notes	3-256
7.6 Explanation of Information Provided	3-257
7.6.1 Schematics	3-257
7.6.2 I/O Model Files	3-272
7.6.3 Board Files	3-272
7.6.4 Bill of Materials	3-272
7.6.5 Photoplot Log	3-272
7.6.6 Netlist Report	3-272
7.6.7 Placed Component Report ..	3-272
7.6.8 Artwork for Each Board Layer	3-272
7.6.9 Trace Segment Line Lengths	3-272
7.6.9.1 Low Addresses (Topology 1)	3-273
7.6.9.2 High Addresses (Topology 4, Point-to-Point) ..	3-275
7.6.9.3 Pentium™ Processor Control (Topology 1)	3-276
7.6.9.4 Pentium™ Processor Control (Topology 3b No 82496)	3-277
7.6.9.5 82496 Control (Topology 3)	3-278
7.6.9.6 82496 Control (Topology 3a Not Connected to Parity 82491's)	3-280

Controller and 82491 Cache SRAM CPU-Cache Chip Set

CONTENTS

PAGE

7.6.9.7 82496 Control (Topology 1b Pentium™ Processor and 82496 Switch Positions)	3-281
7.6.9.8 82496 Control (Topology 4, Point-to-Point) ..	3-282
7.6.9.9 Byte Enables (Topology 5)	3-282
7.6.9.10 CDATA and Parity (Point-to-Point)	3-283
7.6.10 Pentium™ Processor to 82496 Segment Length and Routing Changes	3-285
7.6.11 I/O Simulation Results for Each Net	3-286
7.7 Possible Modification to the Layout	3-289
7.7.1 Non-Parity Layout	3-289
8.0 512K CPU-CACHE CHIP SET OPTIMIZED INTERFACE LAYOUT DESIGN EXAMPLE	3-289
8.1 Layout Objectives	3-290
8.2 Component Placement	3-290
8.3 Signal Routing/Topologies	3-291
8.4 Board/Trace Properties	3-310
8.5 Design Notes	3-311
8.6 Explanation of Information Provided	3-312

ACKNOWLEDGEMENTS

I would like to thank those who have provided their time and expertise to ensure that this document is accurate and complete.

Rob Aslett, Gary Dudeck, Scott Huck, Chris Lane, Dan McCutchan, Prakash Narayanan, Tim Schreyer, and Tawfik Arabi

CONTENTS

PAGE

8.6.1 Schematics	3-312
8.6.2 I/O Model Files	3-326
8.6.3 Board Files	3-326
8.6.4 Bill of Materials	3-326
8.6.5 Photoplot Log	3-326
8.6.6 Netlist Report	3-326
8.6.7 Placed Component Report ..	3-326
8.6.8 Artwork for Each Board Layer	3-326
8.6.9 Trace Segment Line Lengths	3-326
8.6.9.1 Low Addresses and Pentium™ Processor Control	3-327
8.6.9.2 82496 Control	3-331
8.6.9.3 82496 Control	3-333
8.6.9.4 Pentium™ Processor Control	3-334
8.6.9.5 Pentium™ Processor and 82496 Control, High Addresses, Pentium™ Processor Data	3-335
8.6.9.6 Byte Enables	3-338
8.6.9.7 82496 Control	3-340

1.0 INTRODUCTION

The Pentium™ processor, 82496 cache controller, and 82491 cache SRAM CPU-Cache Chip Set has been designed to take advantage of the high performance available from the 66-MHz operating frequency. In addition, it has been designed to obtain this performance without severely adding to the complexity of the system design. These benefits are accomplished by dividing the chip set into two interfaces. The first is the External Interface which is the interface between the CPU-Cache core and the rest of the system. It consists of the memory bus and the memory bus controller. This interface has been designed to operate at a fraction of the CPU's frequency or asynchronous to the CPU. These options simplify the system design by minimizing the portions that must deal with the high frequency signals. The second interface is the Optimized Interface which is between the Pentium processor and the 82496 cache controller and 82491 cache SRAM. This interface is tuned for the known configuration options of the chip set and includes specially designed input and output buffers optimized for the defined electrical environment of each signal path. The specification of the optimized interface is defined to accommodate the tuning involved.

The purpose of this application note is to provide additional explanation of the steps involved in completing a chip set design. It includes:

1. Describing the specifications and how they should be used.
2. Describing Intel's I/O buffer models.
3. Describing the characteristics of printed circuit boards and transmission lines.
4. Stepping through an example of using the information and tools to complete the layout of one signal and verify that it meets the published chip set specifications.

In addition, all of the information associated with a completed chip set optimized interface design example is included. This information allows system designers to minimize their effort in implementing the same or similar design.

2.0 A/C SPECIFICATIONS OF THE CHIP SET

The AC/DC specifications for the chip set are published in the latest revision of the *Pentium™ Processor User's Manual Vol. 2: 82496 Cache Controller and 82491 Cache SRAM Data Book*. It includes the specifications for both the optimized and external interfaces.

2.1 Optimized Interface

The optimized interface is the high-performance interconnect between the Pentium processor and the 82496 cache controller and 82491 cache SRAM. The input and output buffers have been tuned for the defined configuration and electrical environment (loading, etc.). This tuning is what allows the chip set's CPU-Cache core to operate synchronously at 66 MHz.

There are two types of specifications for signals in the optimized interface. The first is Flight Time which is used to guarantee that signal timings are met. The second is Signal Quality to guarantee reliable operation. I/O buffer models have been provided as a tool to ensure these specifications are met. In this section, flight time and signal quality will be discussed. I/O buffer models will be left for the next section.

2.1.1 FLIGHT TIME SPECIFICATION

2.1.1.1 Purpose of Flight Time Specification

The purpose of the flight time specification is to guarantee that a signal supplied by a driving component is available at the receiving component for sampling.

It replaces the output valid delay and input setup time. The two methods are analogous, except that flight time allows the input and output buffer behavior to be matched without major impact to designers or the documentation. In other words, if component A's output is slower than expected, but component B's input is faster than expected, these two can be traded off without having to change the flight time specification. However, if output valid delay and input setup time specifications had been used the specifications would have to be changed. Note that in both cases the time available to the system designer to move the signal from one component to the other is the same.

2.1.1.2 Definition of Flight Time

Flight Time is the propagation delay of a signal from a driving component to any receiving component. It is defined as the time difference between the $V_{cc}/2$ (50%) level of an unloaded output signal and the $V_{cc}/2$ (50%) level of a receiving signal whose 50% V_{cc} to 65% V_{cc} rise time is greater than or equal to 1V/ns. Figure 1 shows the flight time measurement between the 50% V_{cc} points on the unloaded driver and receiver waveforms.

If the rise time between the 50% V_{CC} and 65% V_{CC} points is less than 1V/ns, the determination of flight time is slightly more difficult and requires more calculation. In this case the 65% V_{CC} point is extrapolated

back to the 50% V_{CC} point using a 1V/ns reference slope (i.e., subtract 0.75 ns when $V_{CC} = 5V$). Figure 2 shows the extrapolation from the 65% V_{CC} point and the resulting flight time measurement.

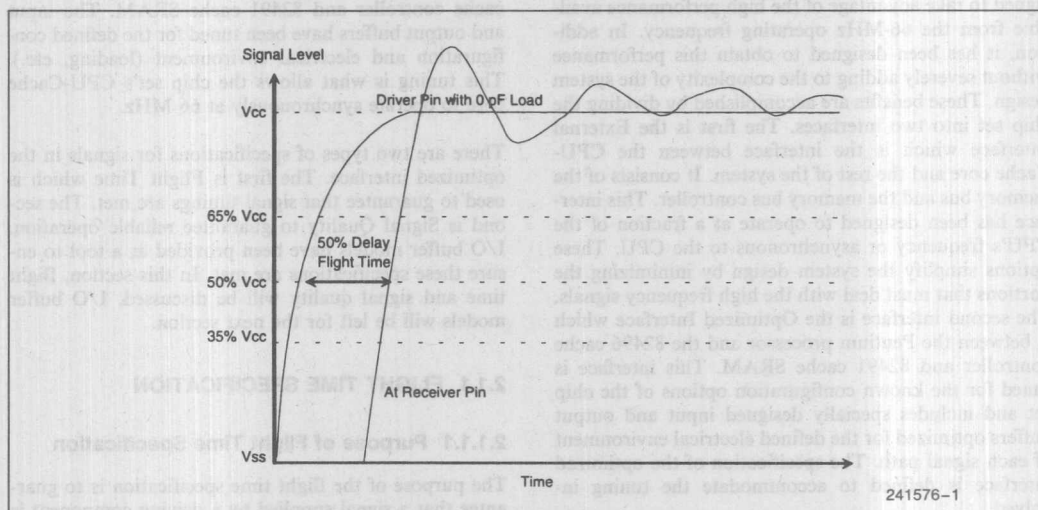


Figure 1. Flight Time Measurement

is required the output valid delay and flight time. The two methods are analogous except the flight time allows the input and output buffer behavior to be matched without input or output or the delay measurement in other words a response to a signal is slower than expected but consistent. A flight time is then expected, these two can be tested off without having to change the flight time specification. However, if output valid delay and input setup time specifications had been used the specification would have to be changed. Note that in both cases the time available to the system designer to move the signal from one component to the other is the same.

2.1.1.3 Definition of Flight Time

Flight Time is the propagation delay of a signal from a driving component to any receiving component. It is defined as the time difference between the Vcc(50%) level of an unloaded output signal and the Vcc(50%) level of a receiving signal where Vcc(50%) is 0.5V. Figure 1 shows the flight time measurement between the 50% Vcc points on the unloaded driver and receiver waveforms.

The purpose of this application note is to provide a model explanation of the steps involved in completing a signal design. It includes:

1. Describing the specifications and how they should be used.
2. Describing Intel's I/O buffer models.
3. Describing the characteristics of printed circuit boards and transmission lines.
4. Describing how to use the information and tools to complete the layout of one signal and verify that it meets the published chip set specifications.

In addition, all of the information associated with a signal design that is not included in the design example is included. This information allows system designers to minimize their effort in implementing the same or similar design.

2.0 I/O SPECIFICATIONS OF THE CHIP SET

The I/O specifications for the chip set are published in the latest revision of the Pentium® Processor User's Manual, Vol. 2: I/O and Cache Controller, and Intel's Cache 241W Data Book. It includes the specifications for both the optimized and standard interfaces.

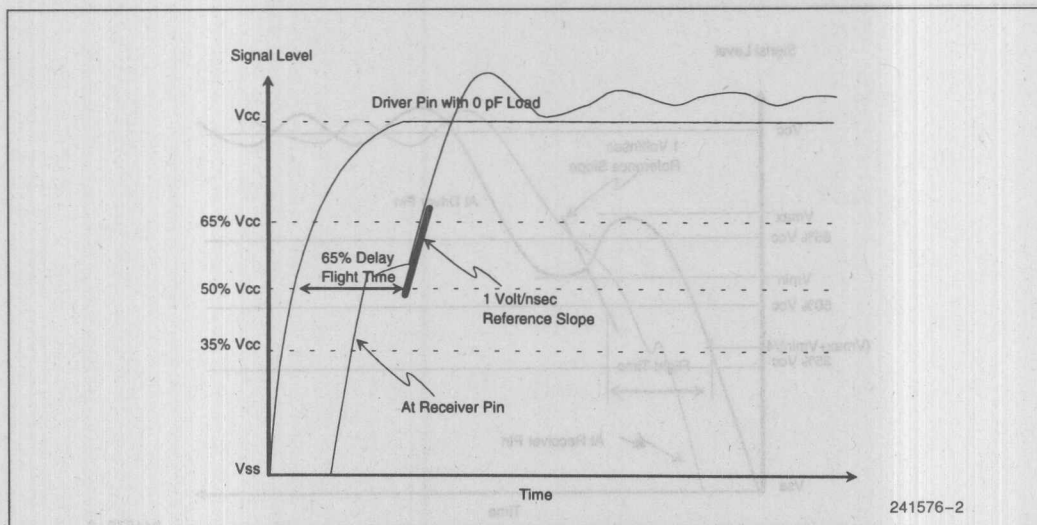


Figure 2. Flight Time Extrapolated from 65% Point

Thus flight time is the longer of T50-50 or Textrapolated. Although Figure 1 and 2 only show low-to-high transitions, flight time is the worst case of low-to-high and high-to-low transitions. Note on high-to-low transitions, 65% Vcc is replaced with 35% Vcc.

In a system environment it will not usually be possible to measure the delay of an unloaded driver. Figure 3 shows the method for measuring flight time in a system environment. As shown, the voltage measured at the pin of the loaded driver will have a ledge near the center of the transition. According to Transmission Line Theory, the time required to reach half the voltage level of the ledge is equivalent to the time required for an unloaded driver to reach the 50% Vcc level. The oscillation (if any) seen at the ledge defines the measurement uncertainty for this technique.

To measure flight time via this technique, first measure the maximum and minimum voltages of the ledge and take the average of these two values, $(V_{max} + V_{min})/2$, to arrive at the ledge voltage. Finally, divide the ledge voltage by two, $(V_{max} + V_{min})/4$. The result is the voltage level that approximately corresponds to the point in time at which an unloaded driver's signal would reach the 50% Vcc level. The flight time is determined by measuring the difference in time between the $(V_{max} + V_{min})/4$ point and the extrapolated 50% point on the receiver, Textrapolated. The uncertainty of this technique is the time difference between the $V_{min}/2$ and $V_{max}/2$ points.

NOTE:

Figure 3 uses Textrapolated. This assumes the rise time between 50% Vcc and 65% Vcc is less than 1V/ns. If the rise time is equal to or greater than 1V/ns, the flight time should be measured to the 50% Vcc point, T0-50.

3

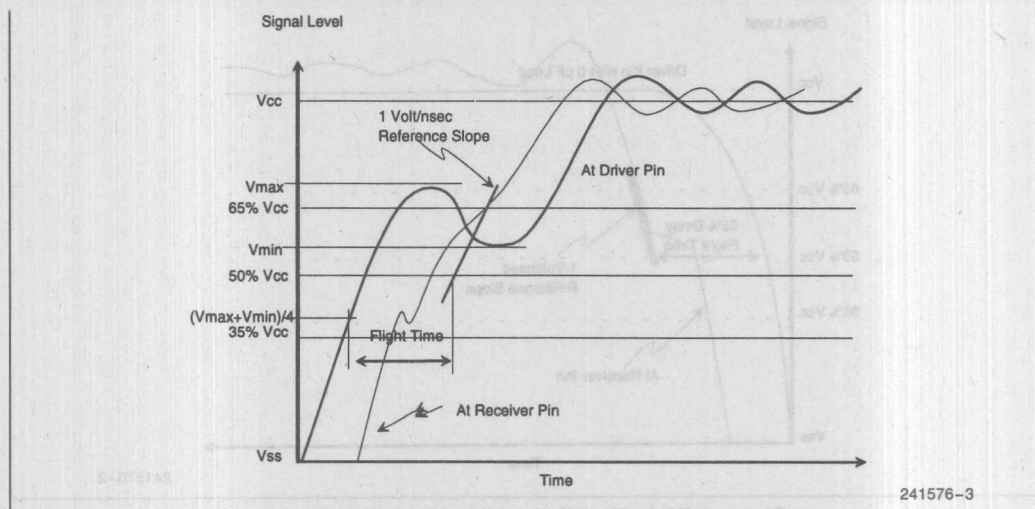


Figure 3. In-System Measurement of Flight Time

2.1.1.3 Clock Skew

Clock skew has generally been included in system design timing analysis. However, as frequencies increase, controlling clock skew becomes more important. Clock skew is the difference in time of the clock signal arriving at different components. It is measured at 0.8V, 1.5V, and 2.0V.

In synchronous devices, the clock signal defines the point in time in which signals are driven or sampled. It is important that all devices have a common reference. If the reference varies from component to component, the difference must be accounted for to ensure the devices function properly. In other words, clock skew must be subtracted from the clock period when performing a timing analysis of a system.

In the CPU-Cache Chip Set, the maximum clock skew between components in the optimized interface is a specification. This specification is required as a complement of flight time to ensure proper functionality. If clock skew exceeds the specified limit, the excess must

be subtracted from the available flight time or the clock period must be increased.

2.1.2 SIGNAL QUALITY SPECIFICATIONS

Acceptable signal quality must be maintained over the entire operating range to insure reliable operation of the chip set. Signal quality consists of four parameters: Over/Undershoot, Time Beyond Supply, Ringback, and Settling Time. Figure 4 illustrates these signal quality parameters and how each is measured for a low-to-high transition. In addition to the absolute maximums associated with individual signals, each of the signal groups defined in the specification must meet maximum group average of Over/Undershoot and Time Beyond Supply. The following sections explain each of these in more detail.

Reliable operation means the signals are sampled correctly, do not exhibit false transitions, and that the long-term reliability of the component is not effected by overdriving the inputs.

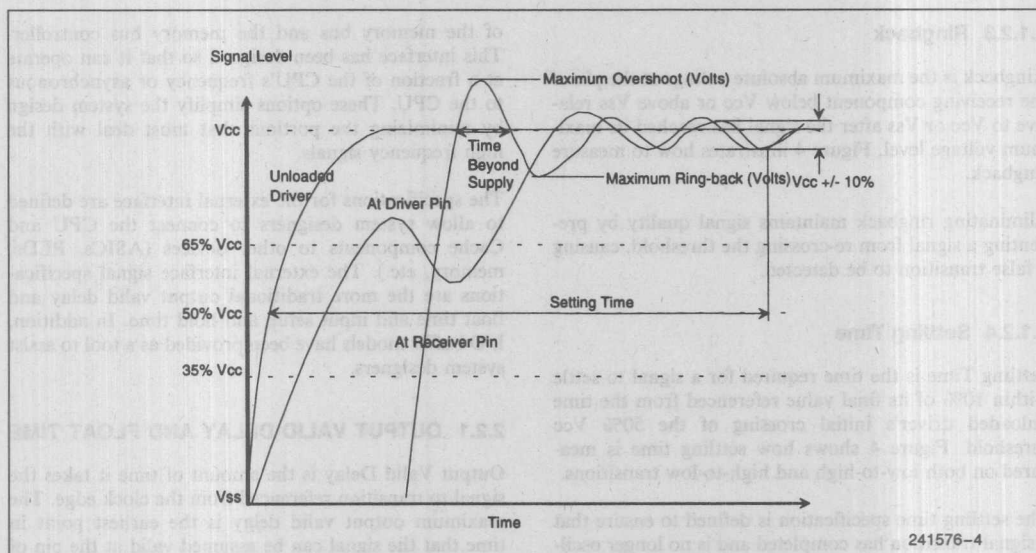


Figure 4. Signal Quality Parameter Measured for Low-to-High Transitions

2.1.2.1 Overshoot

Overshoot is the maximum absolute voltage a signal extends above Vcc or below Vss at the pin of the receiving component. Figure 4 shows the above Vcc case of overshoot. The overshoot specification is defined for use in simulation and assumes that input diodes are not included in the input model during simulation.

The overshoot specification maintains signal reliability by limiting the amount of energy that is injected into the component. Excessive energy being driven into the component can cause both short-and long-term reliability problems. They include Vcc or Vss plane shifts, electromigration, excessive ringback when the input diodes turn off, etc.

2.1.2.2 Time Beyond Supply

Time beyond supply is the maximum time a signal exceeds Vcc or Vss at the pin of the receiving component as shown in Figure 4. If the overshoot voltage is less than or equal to 0.5V, time beyond supply can be ignored. When time beyond supply is being ignored, a value of 0 ns should be used for that signal in calculating the group average time beyond supply.

Time beyond supply is a complement to overshoot when looking at signal quality. The time the signal is beyond the supply is a second factor in limiting the energy driven into the component. By limiting the time beyond supply, system designers are avoiding or minimizing the risk of the same reliability issues as described in the section on overshoot.

2.1.2.3 Ringback

Ringback is the maximum absolute voltage at the pin of the receiving component below V_{CC} or above V_{SS} relative to V_{CC} or V_{SS} after the signal has reached its maximum voltage level. Figure 4 illustrates how to measure ringback.

Eliminating ringback maintains signal quality by preventing a signal from re-crossing the threshold, causing a false transition to be detected.

2.1.2.4 Settling Time

Settling Time is the time required for a signal to settle within 10% of its final value referenced from the time unloaded driver's initial crossing of the 50% V_{CC} threshold. Figure 4 shows how settling time is measured on both low-to-high and high-to-low transitions.

The settling time specification is defined to ensure that a signal transition has completed and is no longer oscillating prior to the next transition. This is important to avoid forcing a signal to transition a distance significantly greater than $V_{CC}/2$. For example, if a signal is still not settled at the time of its next transition, it may be at a voltage above V_{CC} such as 6.5V. Assuming $V_{CC} = 5V$, the transition requires the voltage to swing from 6.5V (instead of 5V) to 2.5V. This added voltage distance translates into added flight time.

2.1.2.5 Group Averages

A Maximum Group Average specification is defined for Overshoot and Settling Time for each signal group. The group average is calculated by summing the maximum overshoot level or settling time for each signal in the group and dividing by the number of signals in the group.

The purpose of the group average specifications is to limit the amount of energy driven into the device. While each input can handle a certain amount of energy as limited by the Overshoot and Time Beyond Supply specifications, the overall part or portions of the part also have limits. These limits are maintained by ensuring the energy driven into these portions does not exceed a certain level.

2.2 External Interface

The External Interface is the interface between the CPU-Cache core and the rest of the system. It consists

of the memory bus and the memory bus controller. This interface has been designed so that it can operate at a fraction of the CPU's frequency or asynchronous to the CPU. These options simplify the system design by minimizing the portions that must deal with the high frequency signals.

The specifications for the external interface are defined to allow system designers to connect the CPU and Cache components to other devices (ASICs, PLDs, memory, etc.). The external interface signal specifications are the more traditional output valid delay and float time and input setup and hold time. In addition, I/O buffer models have been provided as a tool to assist system designers.

2.2.1 OUTPUT VALID DELAY AND FLOAT TIME

Output Valid Delay is the amount of time it takes the signal to transition referenced from the clock edge. The maximum output valid delay is the earliest point in time that the signal can be assumed valid at the pin of the device. This timing is referenced from the clock edge and is measured at 1.5V as illustrated in Figure 5.

NOTE:

This specification is defined assuming $C_L = 0$ pF; therefore, system designers must account for all delay added by the signal traveling to the receiving device.

The maximum output valid delay is used by system designers to perform a worst case timing analysis to ensure that setup times are met at receiving devices.

The minimum output valid delay is the earliest valid data from the previous clock will begin transition after the clock edge. This timing is also referenced from the clock edge and is measured at 1.5V as illustrated in Figure 5.

The minimum output valid delay is used by system designers to perform a worst case timing analysis to ensure that hold times are met at receiving devices. In addition, on I/O or tristate pins, it is used to ensure no bus contention exists due to multiple devices driving the bus simultaneously.

The maximum Output Float Time is the amount of time it takes to float a signal that was driven in the previous clock. This timing is also referenced from the clock edge and is illustrated in Figure 6. Note that the minimum valid delay determines how long data from the previous clock remains valid as shown in Figure 6.

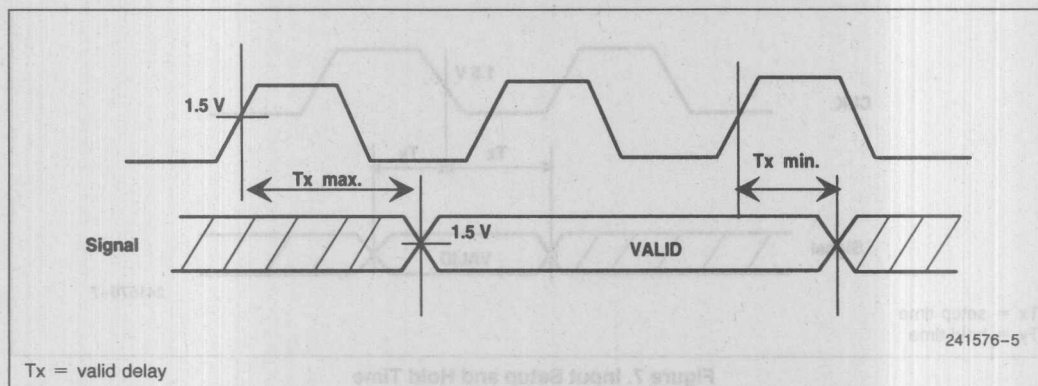


Figure 5. Output Valid Delay

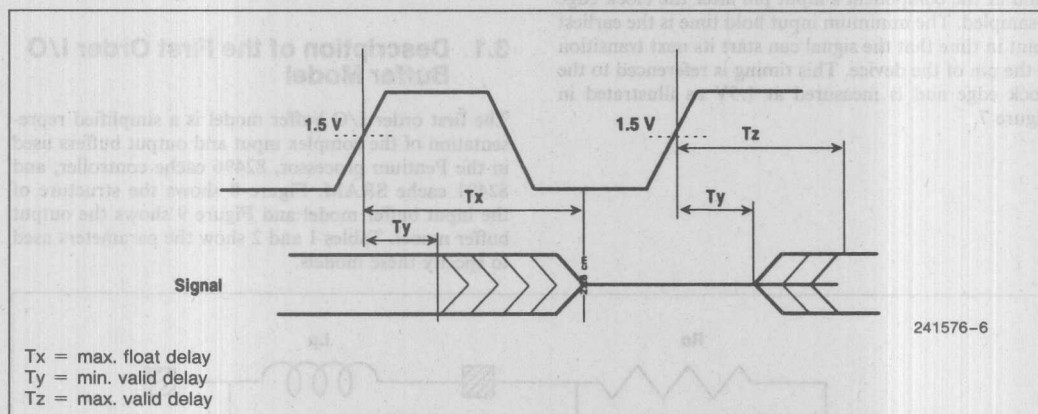


Figure 6. Output Float Time

2.2.2 INPUT SETUP AND HOLD TIME

Input Setup Time is the amount of time a signal must be valid at the component's input pin prior to the clock edge it is sampled. The minimum input setup time is the latest point in time that the signal can be assumed valid at the pin of the device. This timing is referenced to the clock edge and is measured at 1.5V as illustrated in Figure 7.

The input setup time is used by system designers to perform a worst case timing analysis to verify that fast enough drivers have been chosen for ASICs or other devices and that the signals are able to travel across the board layout in the allotted amount of time.

Parameter	Description
C_i	Minimum and maximum value of the capacitance of the input buffer model
L_p	Minimum and maximum value of the package inductance
C_p	Minimum and maximum value of the package capacitance

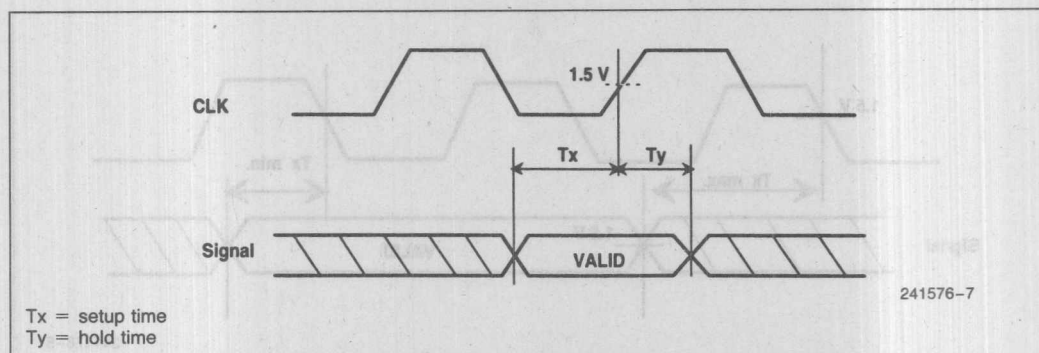


Figure 7. Input Setup and Hold Time

Input Hold Time is the amount of time a signal must be valid at the component's input pin after the clock edge is sampled. The minimum input hold time is the earliest point in time that the signal can start its next transition at the pin of the device. This timing is referenced to the clock edge and is measured at 1.5V as illustrated in Figure 7.

3.0 I/O BUFFER MODELS

3.1 Description of the First Order I/O Buffer Model

The first order I/O buffer model is a simplified representation of the complex input and output buffers used in the Pentium processor, 82496 cache controller, and 82491 cache SRAM. Figure 8 shows the structure of the input buffer model and Figure 9 shows the output buffer model. Tables 1 and 2 show the parameters used to specify these models.

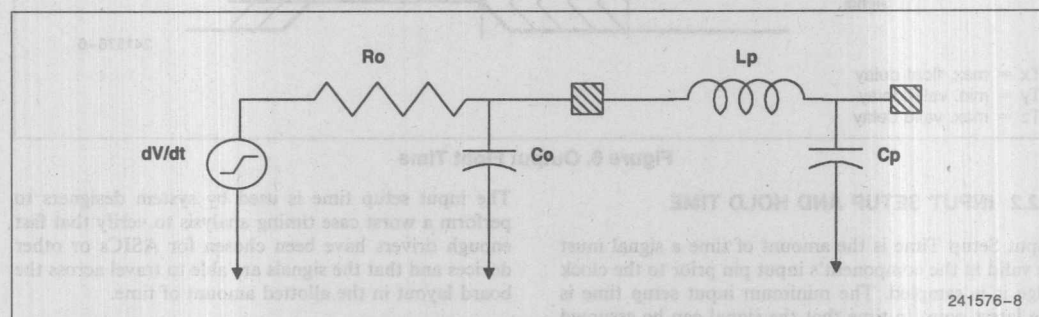


Figure 8. First Order Input Buffer

Table 1. The Parameters Used in the Specification of the First Order Input Buffer Model

Parameter	Description
C_{in}	Minimum and maximum value of the capacitance of the input buffer model
L_p	Minimum and maximum value of the package inductance
C_p	Minimum and maximum value of the package capacitance

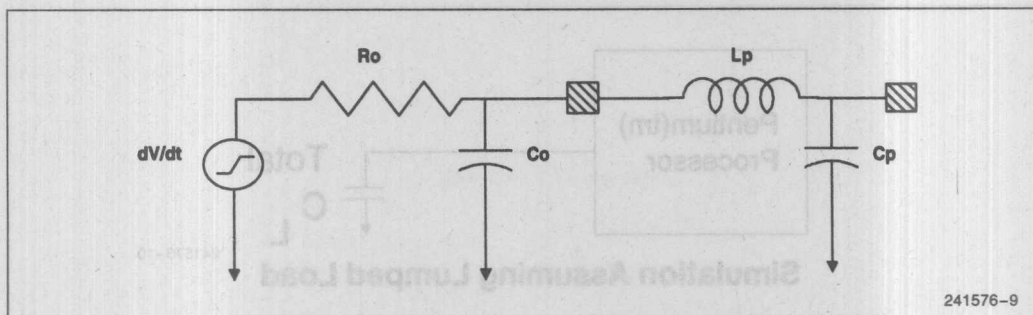


Figure 9. First Order Output Buffer

Table 2. The Parameters Used in the Specification of the First Order Output Buffer Model

Parameter	Description
dV/dt	Minimum and maximum value of the rate of change of the open circuit voltage source used in the output buffer model
Ro	Minimum and maximum value of the output impedance of the output buffer model
Co	Minimum and maximum value of the capacitance of the output buffer model
Lp	Minimum and maximum value of the package inductance
Cp	Minimum and maximum value of the package capacitance

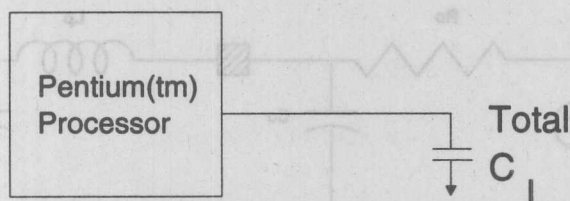
The first order I/O buffer parameters for the chip set are published in the latest revision of the *Pentium™ Processor Data Book*. It includes the minimum and maximum values for each parameter allowing simulations for both the fast and slow condition to be performed.

The key to the first order model is that it is a simplistic representation of the input and output buffers. The parameters are easy to use in a variety of simulators and provide the needed accuracy to complete a chip set design. In addition, the simplicity greatly reduces the compute time required to perform simulations as compared to full transistor and process models.

4.0 HIGH FREQUENCY DESIGN CONSIDERATIONS

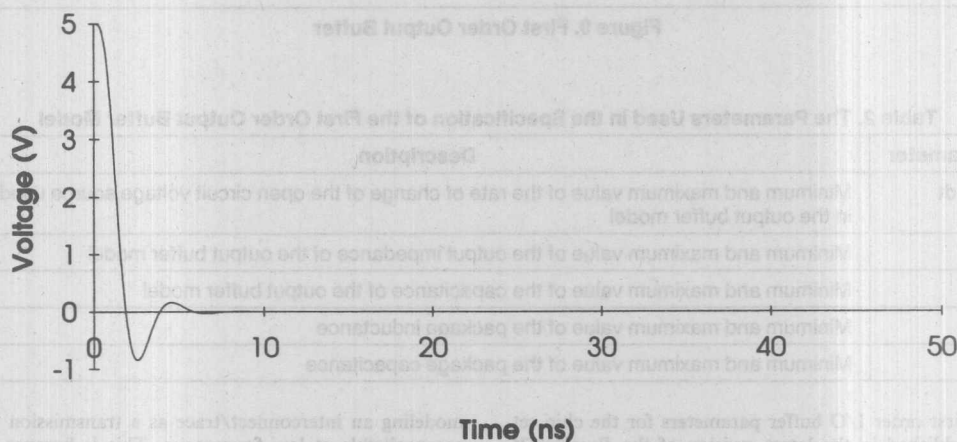
Any board interconnection is a transmission line by definition. However, as a general rule, the effects of

modeling an interconnect/trace as a transmission line are negligible at low frequencies. This is because the reflections get masked since the propagation is short with respect to the signal rise time. In this case, system interconnects can be modelled as lumped loads with no sacrifice to accuracy. As the frequency at which signals change increases, the rise time becomes shorter and transmission line properties become significant and must be considered. Reflections, interference, and noise cause measureable changes in the appearance or behavior of signals at the higher frequencies. They can slow the transition time or cause signal quality violations. Figures 10 and 11 illustrate the effect of modeling traces as lumped loads or as transmission lines. Figure 10 shows a block diagram of a lumped load model and the resulting simulation and Figure 11 shows a block diagram of a transmission line model and the resulting waveform. The transmission line case shows the effects of reflections and how the signal varies at each component. These details are not shown in the lumped load case.



Simulation Assuming Lumped Load

241576-10



241576-11

Figure 10. Lumped Load Simulations

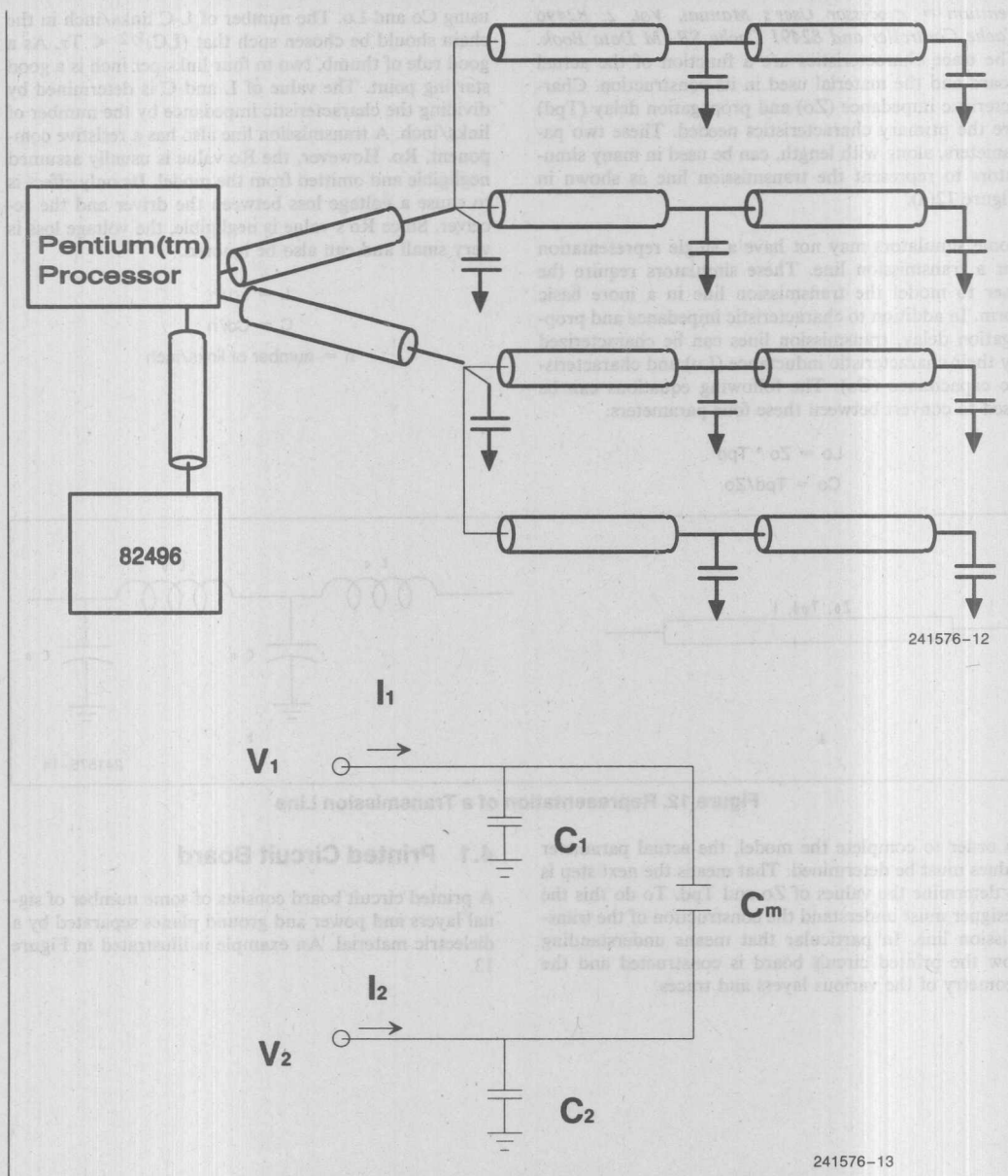


Figure 11. Transmission Line Simulation

To predict a trace's behavior and eliminate the negative effects, it is important to properly model board interconnects as transmission lines. A model consists of the

driving component's output buffer model, the characteristics of the trace, and the receiving component's input buffer model. The I/O buffers are modelled using

the parameters published in the latest revision of the *Pentium™ Processor User's Manual, Vol. 2: 82496 Cache Controller and 82491 Cache SRAM Data Book*. The trace characteristics are a function of the actual board and the material used in its construction. Characteristic impedance (Z_o) and propagation delay (T_{pd}) are the primary characteristics needed. These two parameters, along with length, can be used in many simulators to represent the transmission line as shown in Figure 12(a).

Some simulators may not have a single representation for a transmission line. These simulators require the user to model the transmission line in a more basic form. In addition to characteristic impedance and propagation delay, transmission lines can be characterized by their characteristic inductance (L_o) and characteristic capacitance (C_o). The following equations can be used to convert between these four parameters:

$$L_o = Z_o * T_{pd}$$

$$C_o = T_{pd}/Z_o$$

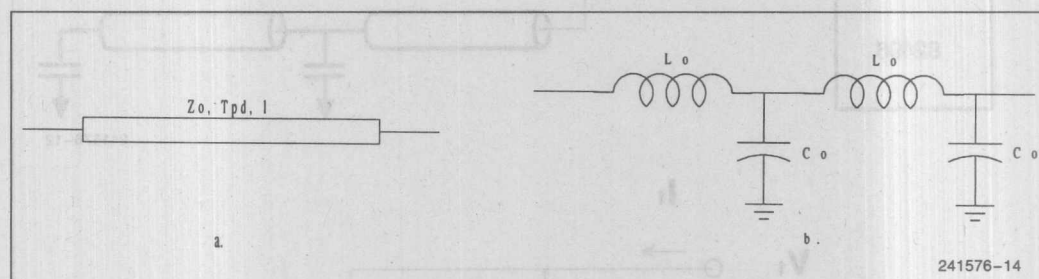


Figure 12. Representation of a Transmission Line

In order to complete the model, the actual parameter values must be determined. That means the next step is to determine the values of Z_o and T_{pd} . To do this the designer must understand the construction of the transmission line. In particular that means understanding how the printed circuit board is constructed and the geometry of the various layers and traces.

Figure 12(b) illustrates the model of a transmission line using C_o and L_o . The number of L-C links/inch in the chain should be chosen such that $(LC)^{1/2} \ll T_r$. As a good rule of thumb, two to four links per inch is a good starting point. The value of L and C is determined by dividing the characteristic impedance by the number of links/inch. A transmission line also has a resistive component, R_o . However, the R_o value is usually assumed negligible and omitted from the model. Its only effect is to cause a voltage loss between the driver and the receiver. Since R_o 's value is negligible, the voltage loss is very small and can also be ignored.

$$L = L_o/n$$

$$C = C_o/n$$

n = number of links/inch

4.1 Printed Circuit Board

A printed circuit board consists of some number of signal layers and power and ground planes separated by a dielectric material. An example is illustrated in Figure 13.

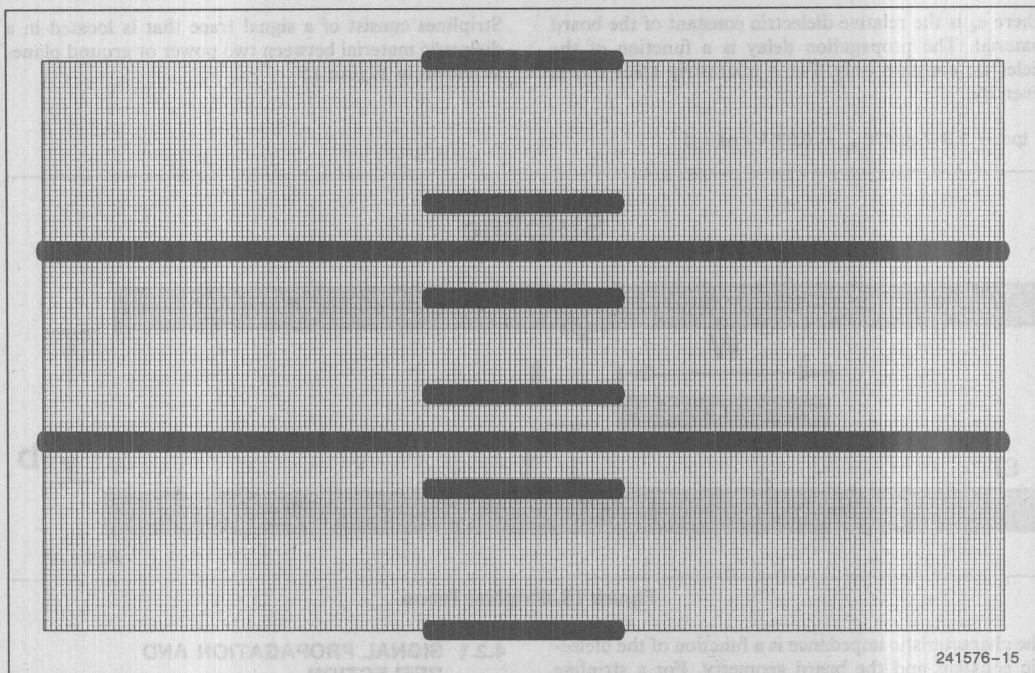


Figure 13. Printed Circuit Board

Although there are many types of transmission lines, those most commonly used on printed circuit boards are microstrip lines and striplines. Microstrip lines con-

sist of a signal trace that is separated from a power or ground plane by a dielectric as shown in Figure 14.

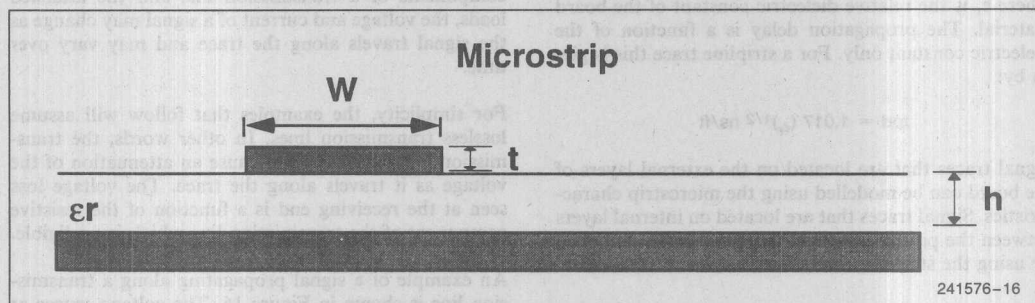


Figure 14. Microstrip Trace

The characteristic impedance is a function of the dielectric constant and the board geometry. For a microstrip trace this is given by:

$$Z_0 = \frac{87}{(\epsilon_r + 1.41)^{1/2}} \ln \left(\frac{5.98h}{0.8W + t} \right) \text{ Ohms}$$

where ϵ_r is the relative dielectric constant of the board material. The propagation delay is a function of the dielectric constant only. For a microstrip trace this is given by:

$$t_{pd} = 1.017 (0.475 \epsilon_r + 0.67)^{1/2} \text{ ns / ft}$$

Striplines consist of a signal trace that is located in a dielectric material between two power or ground planes as shown in Figure 15.

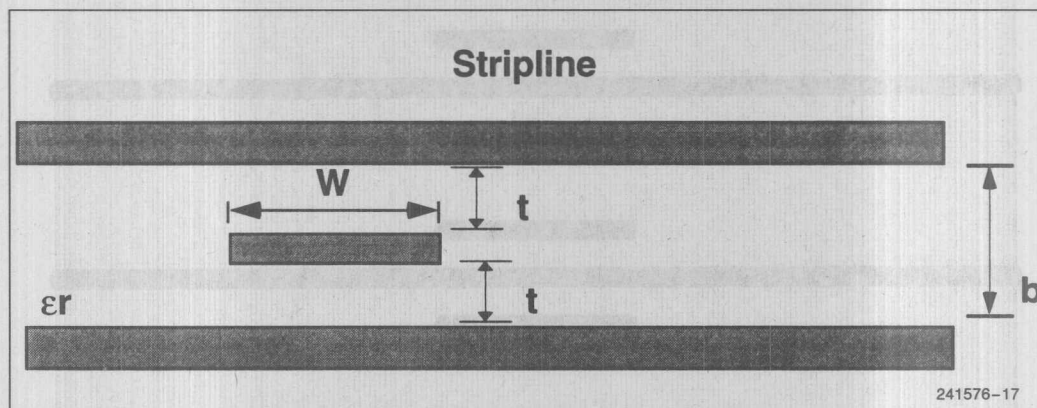


Figure 15. Stripline Trace

The characteristic impedance is a function of the dielectric constant and the board geometry. For a stripline trace this is given by:

$$Z_0 = \frac{60}{(\epsilon_r)^{1/2}} \ln \left(\frac{4b}{0.67 \pi (t + 0.8W)} \right) \text{ Ohms}$$

where ϵ_r is the relative dielectric constant of the board material. The propagation delay is a function of the dielectric constant only. For a stripline trace this is given by:

$$t_{pd} = 1.017 (\epsilon_r)^{1/2} \text{ ns/ft}$$

Signal traces that are located on the external layers of the board can be modelled using the microstrip characteristics. Signal traces that are located on internal layers between the power and ground planes can be modelled by using the stripline characteristics.

4.2 Transmission Line Behavior

Now that the parameters needed to represent a transmission line are complete, the next step is to look at how the transmission line behaves or effects signals that are transmitted on it. The four primary effects are signal propagation and reflection, crosstalk, and skew.

4.2.1 SIGNAL PROPAGATION AND REFLECTION

Ideally, a signal that is driven by one device to another across a trace will reach the receiving device instantly and without any distortion. In reality, this is not the case. Because of the resistive, capacitive, and inductive components of a transmission line and the attached loads, the voltage and current of a signal may change as the signal travels along the trace and may vary over time.

For simplicity, the examples that follow will assume lossless transmission lines. In other words, the transmission line itself does not cause an attenuation of the voltage as it travels along the trace. The voltage loss seen at the receiving end is a function of the resistive component of the transmission line which is negligible.

An example of a signal propagating along a transmission line is shown in Figure 16. The voltage source at the left launches a wave onto the transmission line. The voltage of the wave is equal to the voltage division of the voltage source resistance and the line inductance.

$$V_i = V_{in} * Z_0 / (Z_0 + R_s)$$

This waveform is propagated down the transmission line without any voltage loss or change in the waveform. The only effect of the transmission line is to delay the signal from reaching the receiving end. When the wave reaches the receiving end, it is reflected back towards the source. The reflection is proportional to the initial wave by an amount called the Reflective Coefficient. A reflective coefficient exists for both the source and the load. The values are a function of the source or load resistance and the line's characteristic impedance.

$$\rho_L = (R_L - Z_0)/(R_L + Z_0)$$

$$\rho_S = (R_S - Z_0)/(R_S + Z_0)$$

The value of the initial reflection at the load is:

$$V_r = V_i \cdot \rho_L$$

The reflections can be caused by any discontinuity on the line. The discontinuity can be caused by a mismatch in impedance between the source or load and the characteristic impedance of the trace, branches in the trace, vias, or bends and angles in the trace. Here the discontinuity between the source and load are used as an example because they are probably the most prominent. Each reflection can attenuate or reinforce the wave depending on the phase of the reflection. The reflections continue indefinitely; however, with each reflection the magnitude of the voltage decreases and the line approaches a steady state value. A rule of thumb is that by the third or fourth reflection the value is negligible.

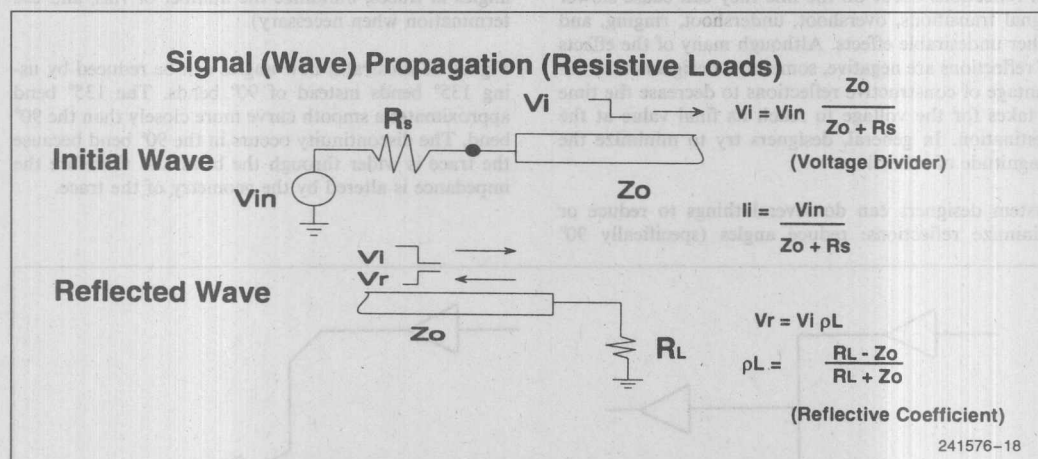


Figure 16. Signal Propagation Along a Transmission Line with Resistive Loads

Now that the voltage of each reflected waveform can be calculated, the next step is to sum these values to determine the voltage measured on the line at any point in time. The superposition principle comes into play. It states that the voltage/current at any point on a transmission line equals the sum of the voltages/currents of all the signals (waves) that have passed that point. In other words, as each reflection passes the point of measurement, it is added to the previous voltage seen at that point. Figure 17 illustrates the voltage seen at the midpoint of the circuit shown in Figure 16 over time.

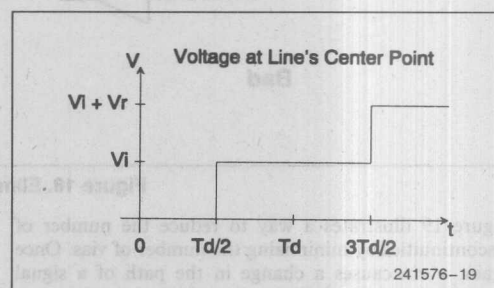


Figure 17. Voltage at the Midpoint of Circuit in Figure 16

From time 0 until $T_d/2$ the voltage is 0V at the midpoint because the initial wave has not yet reached the midpoint. At time $T_d/2$, the initial wave reaches the midpoint and the voltage is V_i . This wave travels down the trace until it reaches the load and a reflection occurs. The reflection begins traveling back towards the

source, but does not reach the midpoint until time $3T_d/2$. At that time the voltage increases by V_r , the reflections voltage as shown in Figure 17. The following equation determines the voltage at any given point on a trace at the given time.

$$V(x,t) = [Z_o / (Z_o + Z_s)] * \{ V_{in} [t - (t - t_{pd} * x)] * U(t - t_{pd} * x) \\ + \rho_L * V_{in} [t - (t - t_{pd} * (2L - x))] * U(t - t_{pd} * (2L - x)) \\ + \rho_L * \rho_S * V_{in} [t - (t - t_{pd} * (2L + x))] * U(t - t_{pd} * (2L + x)) \\ + \rho_L^2 * \rho_S * V_{in} [t - (t - t_{pd} * (4L - x))] * U(t - t_{pd} * (4L - x)) \\ + \rho_L^2 * \rho_S^2 * V_{in} [t - (t - t_{pd} * (4L + x))] * U(t - t_{pd} * (4L + x)) \\ + \dots \}$$

$U(x)$ = unit step function

T_{pd} = propagation delay of signal traveling along the transmission line (ns/ft)

As reflections occur on the line they can cause slower signal transitions, overshoot, undershoot, ringing, and other undesirable effects. Although many of the effects of reflections are negative, sometimes designers take advantage of constructive reflections to decrease the time it takes for the voltage to reach its final value at the destination. In general, designers try to minimize the magnitude of reflections.

angles in traces, minimize the number of vias, and use termination when necessary).

Figure 18 illustrates how angles can be reduced by using 135° bends instead of 90° bends. The 135° bend approximates a smooth curve more closely than the 90° bend. The discontinuity occurs in the 90° bend because the trace is wider through the bend and therefore the impedance is altered by the geometry of the trace.

System designers can do several things to reduce or minimize reflections: reduce angles (specifically 90°

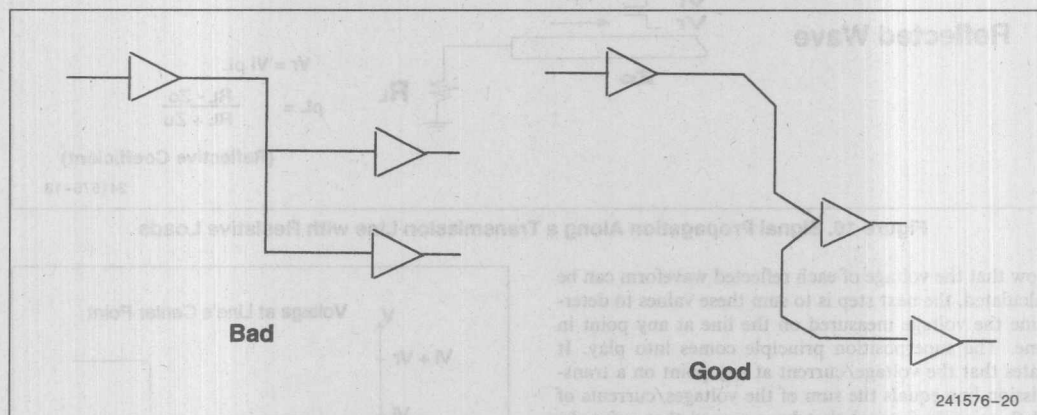


Figure 18. Eliminate 90° Angles

Figure 19 illustrates a way to reduce the number of discontinuities by minimizing the number of vias. Once again, a via causes a change in the path of a signal much like the 90° angles do. In addition, the geometry of a via is generally wider or thicker than the rest of

the trace resulting in a different impedance for that portion of the interconnect. The change of impedance in the path causes the discontinuity and the resulting reflections.

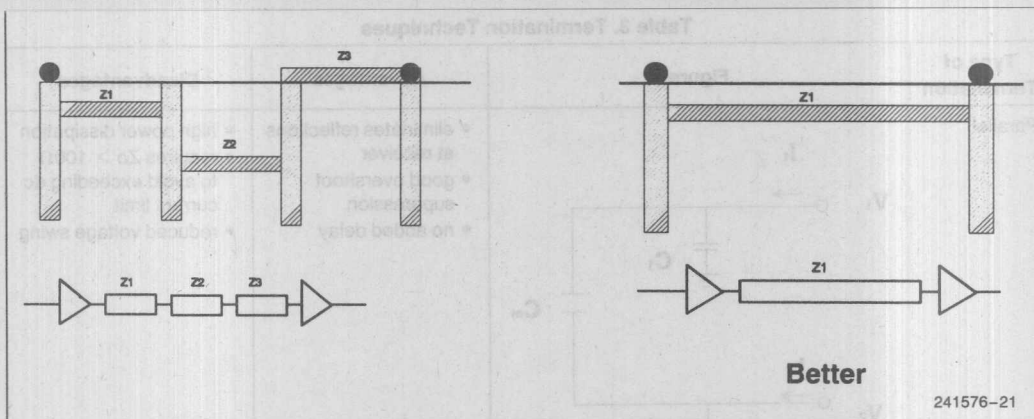


Figure 19. Minimize Vias

It is not always possible to eliminate all the discontinuities or mismatches in impedance. When this is the case, it is sometimes necessary to use a technique called *Termination* to artificially make the mismatched impedances appear matched. This technique is normally used to match a traces characteristic impedance with either the load or sources impedance.

$$Z_o = Z_{\text{term}} + Z_L$$

and

$$Z_o = Z_{\text{term}} + Z_S$$

Several techniques of termination exist. They are Parallel, AC or RC, and Series termination. Each technique has its own advantages and disadvantages as summarized in Table 3.

3

Table 3. Termination Techniques

Type of Termination	Figure	Advantages	Disadvantages
Parallel	<p> $R1 \parallel R2 = Z_0$ $R2 = 2.6Z_0$ $R1 = R2/1.6$ </p>	<ul style="list-style-type: none"> eliminates reflections at receiver good overshoot suppression no added delay 	<ul style="list-style-type: none"> high power dissipation requires $Z_0 > 100\Omega$ to avoid exceeding dc current limit reduced voltage swing
AC or RC	<p> $R_{Term} = Z_0$ $R_{Term} C_{Term} = t_{Rise/Fall}$ </p>	<ul style="list-style-type: none"> low power consumption full voltage swing eliminates initial reflection at receiver 	<ul style="list-style-type: none"> C_{term} adds capacitive Load to driver added delay due to RC time constant component size and count
Series	<p> $R_{Term} = Z_0 - R_s$ </p>	<ul style="list-style-type: none"> no additional loading on driver no additional charging time low power consumption eliminates secondary reflection at source 	<ul style="list-style-type: none"> added delay

4.2.2 CROSSTALK

Crosstalk is another side effect of transmission line interconnects. Crosstalk is the result of fields from adjacent traces interacting with each other. The interaction can alter the characteristics of a driven line or cause noise to be coupled into passive lines.

Crosstalk can be characterized by two parameters: Mutual Inductance, L_m , and Mutual Capacitance, C_m , as shown in Figure 20 and 21, respectively. These two

parameters represent the inductive and capacitive values that exist between two adjacent lines. The inductance allows a current in one line to induce a voltage in a second line.

$$V_{m2} = L_m \cdot \Delta I_1 / \Delta t$$

The capacitance allows a voltage on one line to induce a current in the second.

$$I_{m2} = C_m \cdot \Delta(V_1 V_2) / \Delta t$$

These mutual components have an additive effect to the L and C used to characterize each transmission line. To see what effect this has, examine the two components separately. First, Figure 20 illustrates two parallel traces with their inductive components and a mutual inductance between the two.

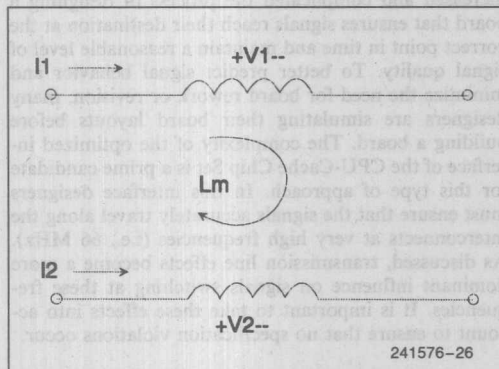


Figure 20. Inductive Components of Two Parallel Traces

The voltage seen on each line is given by:

$$\begin{aligned} V_1 &= V_{L1} + V_m = (L_1 * \Delta I_1 / \Delta t) + (L_m * \Delta I_2 / \Delta t) \\ V_2 &= V_{L2} + V_m = (L_2 * \Delta I_2 / \Delta t) + (L_m * \Delta I_1 / \Delta t) \end{aligned}$$

To see what effect this has assume $L_1 = L_2$ and the magnitude of $\Delta I_1 / \Delta t = \Delta I_2 / \Delta t$. This allows the above equations to be simplified to:

$$\begin{aligned} V_1 &= V_2 = (L_1 + L_m) * \Delta I_1 / \Delta t \\ &\text{(Current in same direction)} \end{aligned}$$

and

$$\begin{aligned} V_1 &= -V_2 = (L_1 - L_m) * \Delta I_1 / \Delta t \\ &\text{(Current in opposite direction).} \end{aligned}$$

From these equations the effective inductance seen on either trace is:

$$\begin{aligned} L_{\text{eff}} &= L_1 + L_m \\ &\text{(Current in same direction)} \end{aligned}$$

and

$$\begin{aligned} L_{\text{eff}} &= L_1 - L_m \\ &\text{(Current in opposite direction).} \end{aligned}$$

Therefore, if the currents are flowing in the same direction the effective inductance of each trace is increased. If the currents are in opposite directions the effective inductance of each trace is decreased.

Secondly, Figure 21 illustrates two parallel traces with their capacitive components and a mutual capacitance between the two.

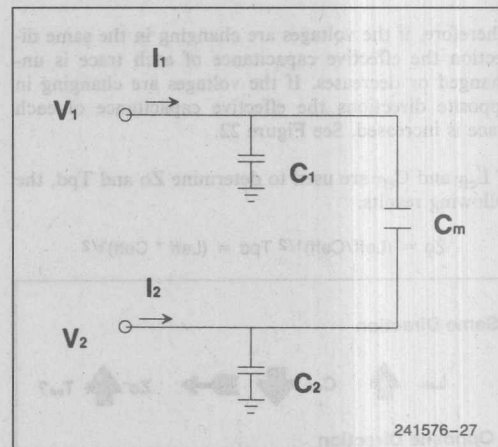


Figure 21. Capacitive Components of Two Parallel Traces

The current seen in each line is given by:

$$\begin{aligned} I_1 &= I_{C1} + I_M \\ &= (C_1 * \Delta V_1 / \Delta t) + (C_m * \Delta(V_1 - V_2) / \Delta t) \\ &= ((C_1 + C_m) * \Delta V_1 / \Delta t) - (C_m * \Delta V_2 / \Delta t) \\ I_2 &= I_{C2} + I_M \\ &= (C_2 * \Delta V_2 / \Delta t) + (C_m * \Delta(V_2 - V_1) / \Delta t) \\ &= ((C_2 + C_m) * \Delta V_2 / \Delta t) - (C_m * \Delta V_1 / \Delta t) \end{aligned}$$

Using the same assumptions that $C_1 = C_2$ and that the magnitude of $\Delta V_1 / \Delta t = \Delta V_2 / \Delta t$ allows the equations to be simplified to:

$$I_1 = I_2 = C_1 * \Delta V_1 / \Delta t \text{ (Voltage change in the same direction on both traces)}$$

and

$$I_1 = -I_2 = (C_1 + 2C_m) * \Delta V_1 / \Delta t \text{ (Voltage change in the opposite direction on each trace).}$$

From these equations the effective capacitance seen on either trace is:

$$C_{\text{eff}} = C_1 \quad (\text{Voltage change in same direction})$$

and

$$C_{\text{eff}} = C_1 + 2C_m \quad (\text{Voltage change in opposite directions}).$$

Therefore, if the voltages are changing in the same direction the effective capacitance of each trace is unchanged or decreases. If the voltages are changing in opposite directions the effective capacitance of each trace is increased. See Figure 22.

If L_{eff} and C_{eff} are used to determine Z_0 and T_{pd} , the following results:

$$Z_0 = (L_{\text{eff}}/C_{\text{eff}})^{1/2} \quad T_{\text{pd}} = (L_{\text{eff}} * C_{\text{eff}})^{1/2}$$

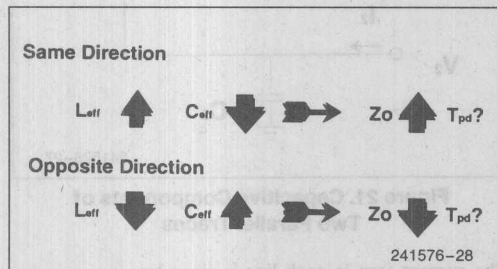


Figure 22. Effect of Changing Voltages in the Same or Opposite Directions

Electrons travel at the speed of light, so T_{pd} can never decrease. Therefore, T_{pd} either remains constant or increases.

This altering of Z_0 and T_{pd} by crosstalk explains why termination is never 100% effective. The crosstalk leads to a variation between the targeted Z_0 and the actual Z_0 . Termination is usually defined to match the targeted Z_0 's. The result is an interconnect that is not perfectly matched via termination.

5.0 CHIP SET DESIGN

As simulation tools have improved it has become easier to test design assumptions before actually spending the time or money to build a printed circuit board. In addition, the frequencies at which signals switch have also increased and complicated the process of designing a board that ensures signals reach their destination at the correct point in time and maintain a reasonable level of signal quality. To better predict signal behavior and minimize the need for board rework or revision, many designers are simulating their board layouts before building a board. The complexity of the optimized interface of the CPU-Cache Chip Set is a prime candidate for this type of approach. In this interface designers must ensure that the signals accurately travel along the interconnects at very high frequencies (i.e., 66 MHz). As discussed, transmission line effects become a more dominant influence on signals switching at these frequencies. It is important to take these effects into account to ensure that no specification violations occur.

A possible scenario for designing the optimized interface is shown in Figure 23. As always the first step is to understand the specifications. This document along with the published specifications should help complete this step. Based on these specifications, system geometry requirements, and an understanding of the board's basic electrical characteristics, a first pass component placement and routing can be completed. Once the routing is complete or possibly as part of the routing, individual traces should be simulated to determine their electrical behavior. This includes examining both flight time and signal quality for each signal and determining if it meets the specification. Any signals that violate the specification must be modified. Portions of this document will provide some information and guidelines on how to modify or route the traces to meet the specifications. With each change, the routing should be re-simulated to ensure the specifications are still met.

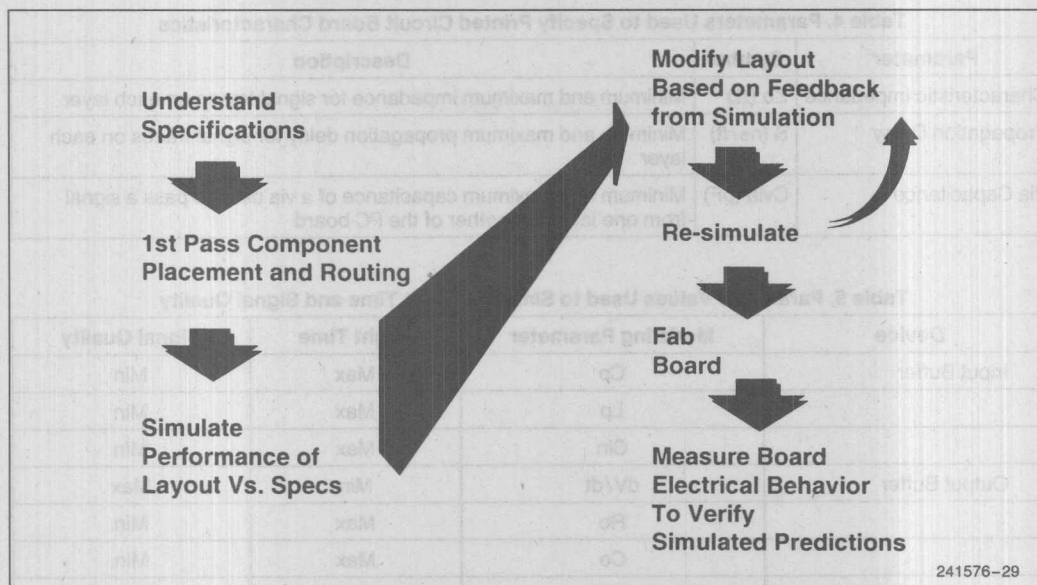


Figure 23. Process for Completing Optimized Interface Design

Once all the specifications are met, it is time to build the board. The goal is that once this board is manufactured and the components are installed, it will meet specification without any changes. However, this must be verified by making actual measurements on the board to verify all of the flight time and signal quality specifications are met. It is also beneficial to make sure that the actual measurements correlate to the predicted results from simulation. This is especially helpful if any corrections are required to bring the board within specification.

The next couple of sections will describe the requirements and guidelines that should be followed while making these simulations and measurements.

5.1 Simulation Environment

The environment chosen to simulate the optimized interface is very critical. A number of different options are available on the market today. It is the system designer's responsibility to select the option best suited for their design requirements. These requirements will in-

clude the accuracy of the results; as well as, how easy it is to import schematics, layout routing, or modeling parameters.

5.1.1 SIMULATION REQUIREMENTS

When simulating the optimized interface to determine flight time and signal quality, it is important that the appropriate modeling parameters are used. The I/O models are provided with minimum and maximum values for each parameter. Using these values the fast and slow corners of the buffer's behavior can be modeled. In addition, the printed circuit board can be modeled for its fast and slow corners. Table 4 restates the characteristics of a printed circuit board.

Flight time is determined by simulating with the slow corner used for all parameters. In this corner signals require the longest amount of time to transition and reach their destination. The fast corner is used to simulate signal quality. In the fast corner, signals transition their fastest and are therefore their noisiest. Table 5 summarizes the parameter values used to simulate for flight time and signal quality.

Table 4. Parameters Used to Specify Printed Circuit Board Characteristics

Parameter	Symbol	Description
Characteristic Impedance	Zo (Ω)	Minimum and maximum impedance for signal traces on each layer
Propagation Delay	S (ns/ft)	Minimum and maximum propagation delay for signal traces on each layer
Via Capacitance	Cvia (pF)	Minimum and maximum capacitance of a via used to pass a signal from one layer to another of the PC board

Table 5. Parameter Values Used to Simulate Flight Time and Signal Quality

Device	Modeling Parameter	Flight Time	Signal Quality
Input Buffer	Cp	Max	Min
	Lp	Max	Min
	Cin	Max	Min
Output Buffer	dV/dt	Min	Max
	Ro	Max	Min
	Co	Max	Min
	Lp	Max	Min
	Cp	Max	Min
	Zo	Min	Max
Printed Circuit Board	S	Max	Min
	Cvia	Max	Min
	Temperature	Max	Min
Other	Vcc	Min	Max

These values should be used to define the simulation model files used to simulate for flight time and signal quality.

While simulating the two corners it should become obvious that there will be trade-offs in optimizing for one or the other. Some sacrifices in signal quality may be required to ensure flight time specifications are met, or vice versa.

5.2 Routing Signal Traces for Their Optimal Performance

Priority should be given to optimizing the performance of the signals in the optimized interface. For the 256K byte layout example that Intel completed, the signals have been divided into the categories listed in Table 6. These categories are based on fanout and connectivity characteristics.

Table 6. Optimized Interface Signal Categories

Category	Signal
Low Address (connected to PP, CC, and CS)	A3–A16, HITM#, W/R#
High Address (connected to PP and CC)	A17–31, BT0–3
PP-CC Control (connected to PP and CC)	Driven by PP: ADSC#, AP, CACHE#, D/C#, LOCK#, M/IO#, PCD, PWT, SCYC Driven by CC: AHOLD, BRDYC1#, EADS#, INV, KEN#, NA#, WB/WT#
PP-CS Control (connected to PP and CSs)	ADS#
CC-CS Control (connected to CC and CSs)	BLAST#, BRDYC2#, BUS#, MAWEA#, MCYC#, WBA, WBTP, WBWE#, WRARR#, WAY
Other CC Control	BLEC#, BOFF#
Byte Enables	BE0#–BE7#
CPU Data and Parity	CD0–CD63, CP0–CP7

PP=Pentium processor
CC=82496 cache controller
CS=82491 cache SRAM

Within each category the routing or topology should be defined to minimize delay while maintaining acceptable signal quality. To do this and maintain the manufacturability of the board, rules were defined to govern the line lengths for each segment of a topology. To develop these rules some analysis of board characteristics and signal behavior is necessary.

5.2.1 RULES FOR OPTIMIZING SIGNAL ROUTING

Both the fast and slow corners must be considered to ensure both flight time and signal quality are met by optimizing a signal's routing.

Flight time is minimized by optimizing each interconnect to minimize the distance the signal must travel and the loading presented to the driver. The dominant opposition to minimizing these factors is the printed circuit board's geometry requirements (i.e., physical distance between components and component placement) and electrical characteristics (propagation delay and characteristic impedance).

The strategy used to optimize each interconnect for signal quality is to make each net's routing electrically symmetric. This is especially important on heavily loaded nets.

Electrically symmetric means the delays of each branch within the net are equal when viewed from the driver. Figure 24 shows a topology from the 256 Kbyte layout example that illustrates this principle. For this topology with the Pentium processor driving, the symmetry is best when the delay from the Pentium processor to the 82496 cache controller is equal to the delay from the Pentium processor to the farthest 82491 cache SRAM. By making these delays equal, the round-trip delays are also equal, and therefore any reflections return to the Pentium processor simultaneously. By returning simultaneously, the reflections can rapidly cancel each other, resulting in the waveform settling quickly.

If these two delays are not equal, asymmetric reflections return to the Pentium processor at different times, and do not cancel each other. The result is a complex interference pattern that generates considerable ringing. In some cases this ringing can last for more than one clock cycle.

5.2.2 DETERMINING THE OPTIMAL NET

There are two methods of optimizing the line lengths and relationships of traces within a net. One uses an asymmetry factor [5] to identify the optimal relationship. The other uses settling time to find this relationship.

Using the asymmetry factor to optimize the symmetry of the net in Figure 24, the input impedance (frequency-domain) of each branch was calculated from the driver's point of view. The branch impedances were compared to each other and the difference was integrated over all

frequencies, resulting in a symmetry energy factor which quantifies the amount of symmetry in the topology. Figure 25 also shows a plot of this factor as a function of the lengths of segments L_a and L_b .

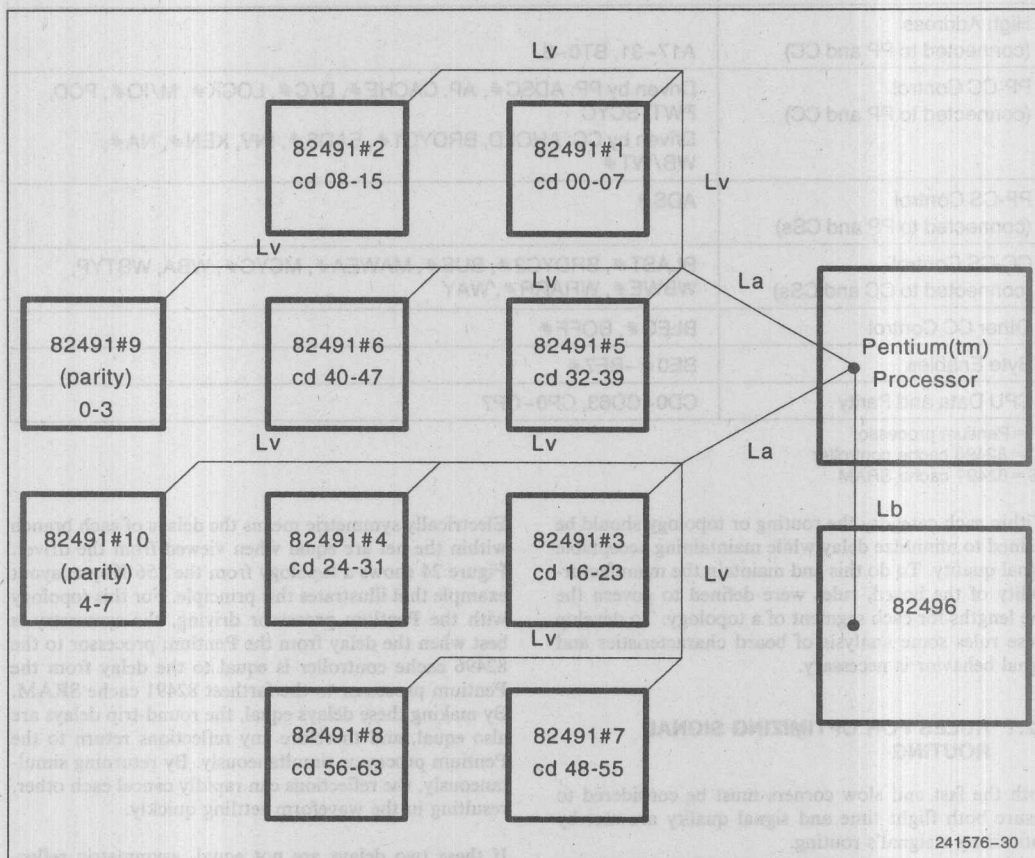


Figure 24. Energy Minimization for a Given Topology

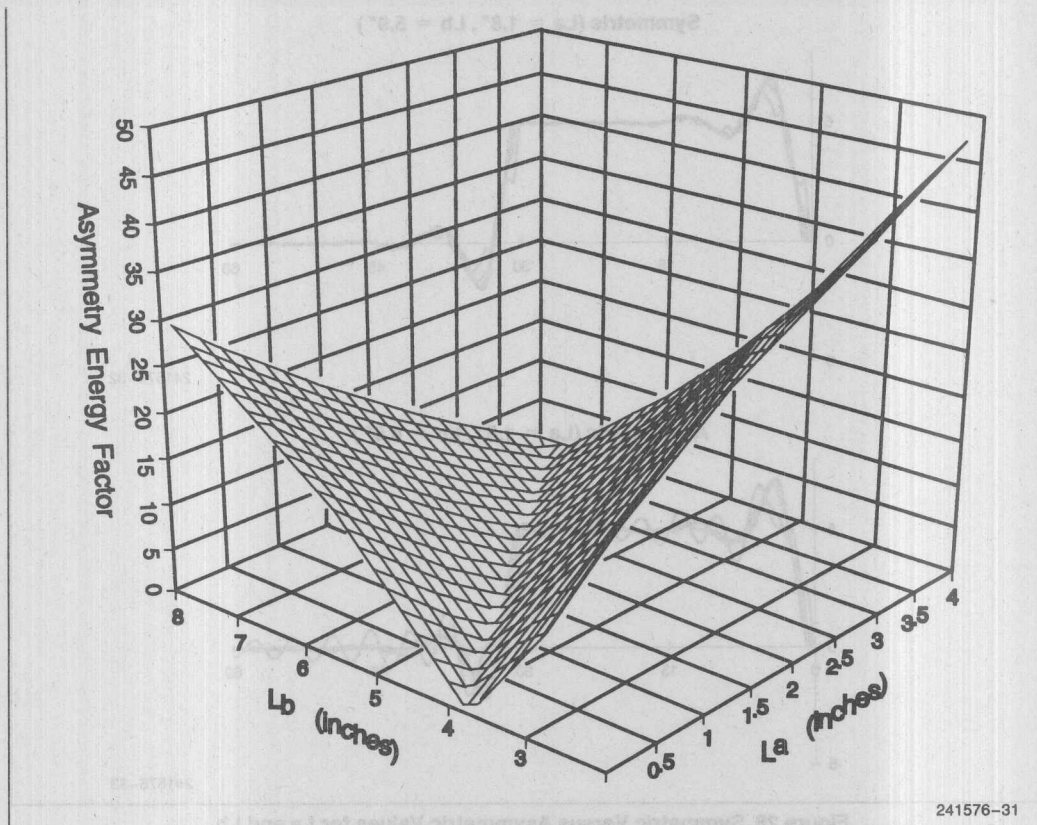


Figure 25. Energy Minimization for a Given Topology

There is a strong correlation between the asymmetry factor and the net's signal quality. This is reinforced by simulating the topology using values for L_a and L_b from the plot in Figure 25. Figure 26 shows the waveforms obtained by simulating the topology with symmetric values for L_a and L_b from along the energy minimum. The line of points in the plot of Figure 25 where the energy minimum occurs corresponds to to-

pologies that are electrically symmetric. An asymmetric topology is obtained by using values for L_a and L_b that lie away from the energy minimum. The waveform obtained by simulating this asymmetric case is also shown in Figure 26. Notice the difference in signal quality in the two plots. The symmetric case is much better than the asymmetric.

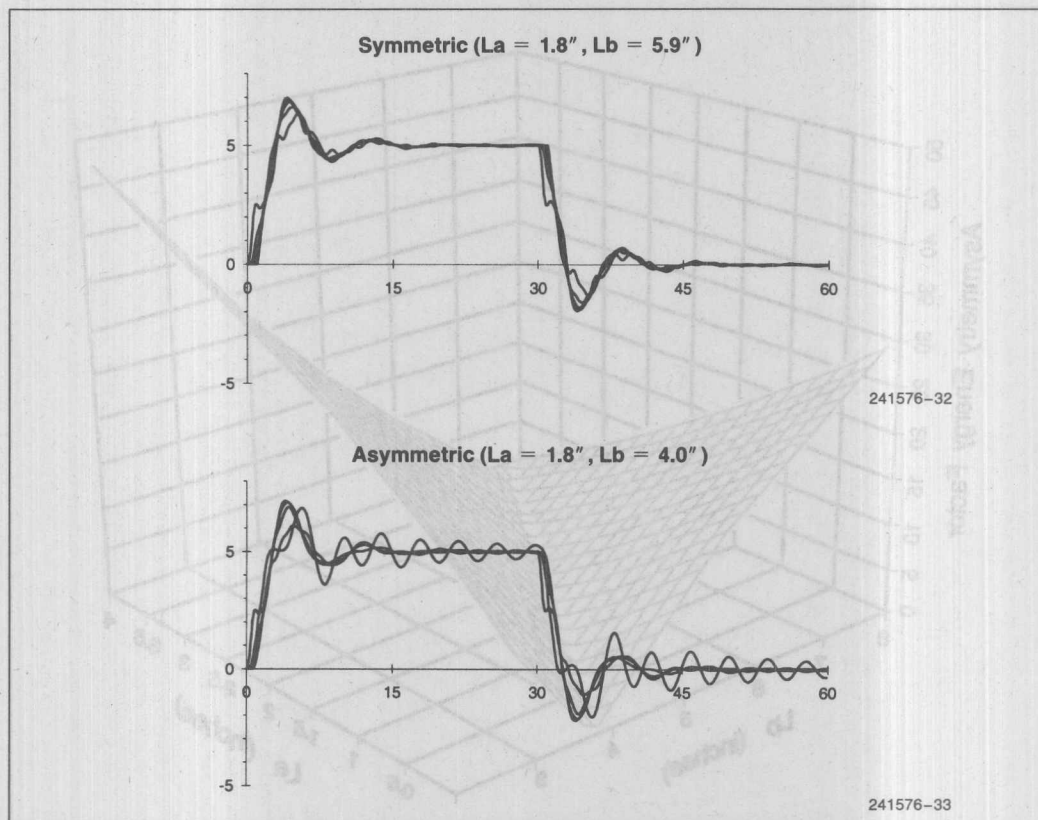


Figure 26. Symmetric Versus Asymmetric Values for L_a and L_b

There is a strong correlation between the asymmetric factor and the net signal density. This is reinforced by simulating the topology using values for L_a and L_b from the data in Figure 25. Figure 26 shows the waveforms obtained by simulating the topology with symmetric values for L_a and L_b from along the energy minimum. The line of points in the plot of Figure 25 where the energy minimum occurs corresponds to 10-

politics that are electrically symmetric. An asymmetric topology is obtained by using values for L_a and L_b that lie away from the energy minimum. The waveform obtained by simulating the asymmetric case is also shown in Figure 26. Notice the difference in signal density in the two plots. The asymmetric case is much better than the symmetric case.

This technique can be used to optimize the routing of all heavily loaded signals in a chip set design. From the energy factor plot rules can be defined to govern the segment lengths needed to minimize the energy factor and obtain the specified signal quality.

The 256K layout example that follows used this technique extensively to route the heavily loaded signals. For each signal group or topology, the asymmetry energy factor was calculated as a function of the topology's segment lengths and a set of rules defined to govern the segment lengths required to provide a routing that meets the signal quality specifications.

Similar results can be determined by using settling time to the optimal routing. To optimize the symmetry of a net, the settling time is plotted against line length. The minimum settling time occurs at the point where the net is balanced. Figure 27 shows a settling time plot for the net in Figure 24. Settling time is plotted against the true length for the segment between the Pentium processor and the 82496 for a given length between the Pentium processor and 82491. For $L_a = 1.8$ inches the settling time approach recommends $L_b = 5.8-6.0$ inches.

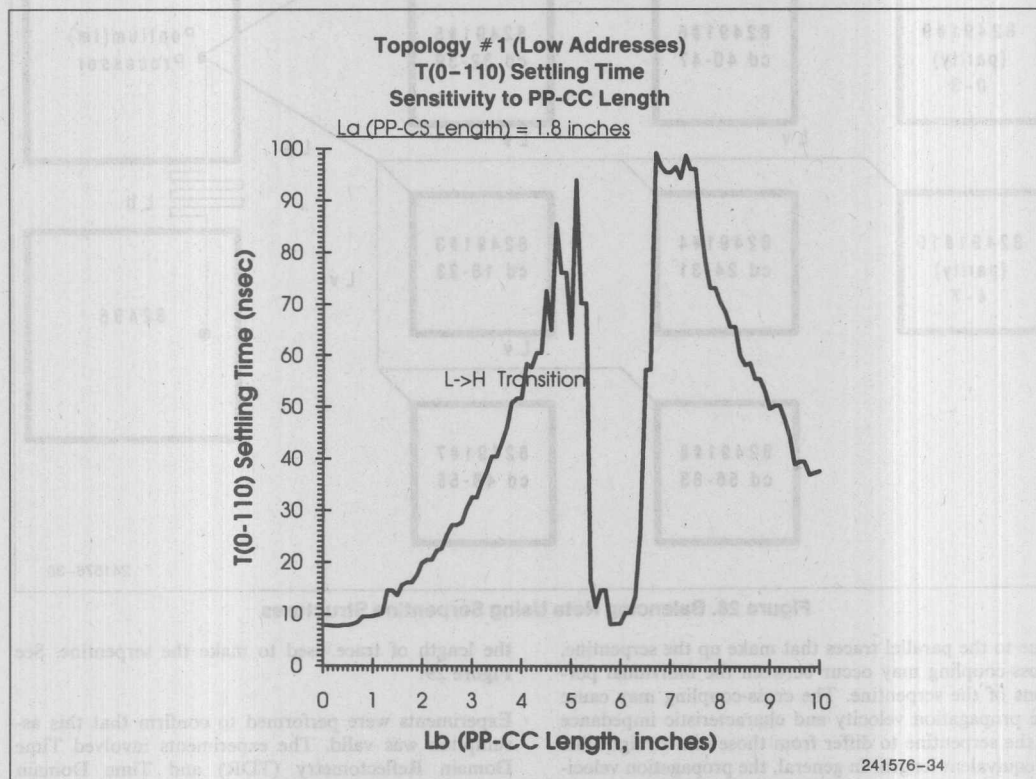


Figure 27. Settling Time Versus Line Length

5.2.3 SERPENTINE STRUCTURES

Serpentine structures are one design technique that can be used to assist in balancing the interconnect delays between the Pentium processor, 82496 cache controller, and 82491 cache SRAM components. The structure is used to add length to specific traces within the nets.

The goal of adding this length is to make the net "balanced" or electrically symmetrical. In particular, this technique has been used to add length to the trace between the Pentium processor and 82496 cache controller so that it is electrically symmetric with the traces between the Pentium processor and 82491s of the same net as shown in Figure 28.

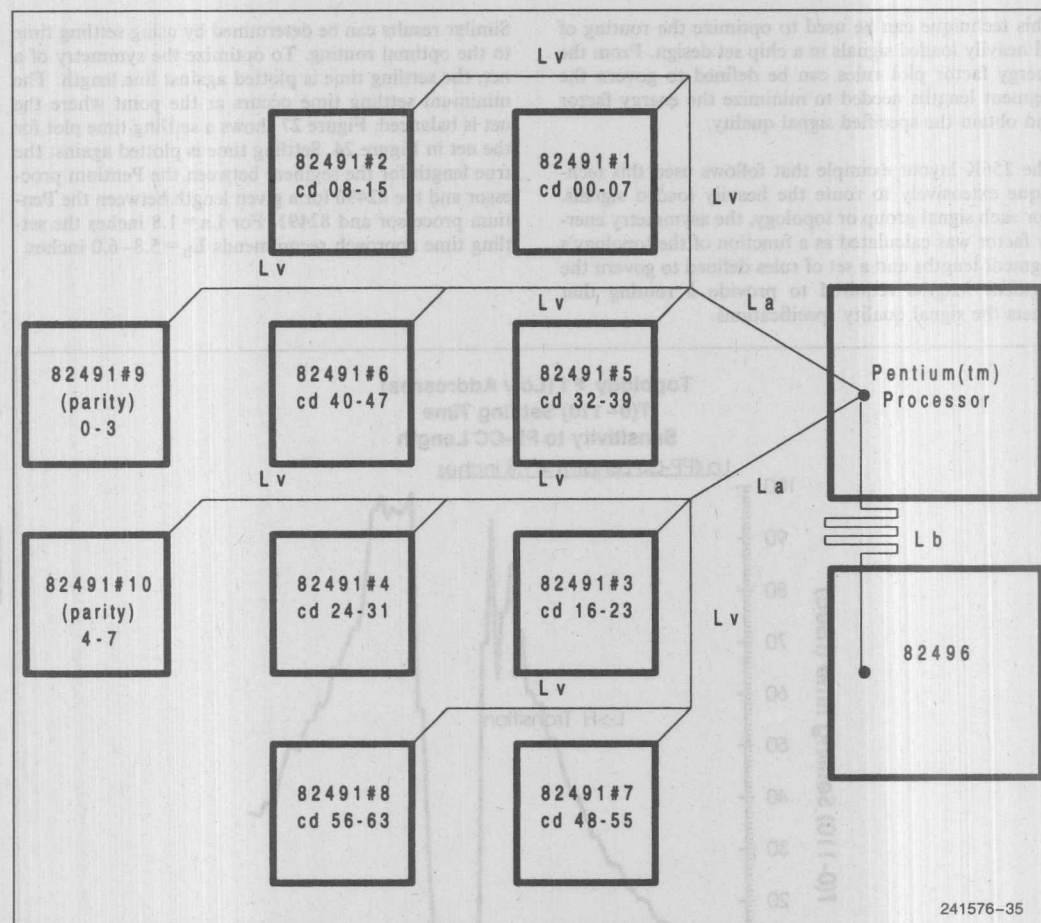


Figure 28. Balancing Nets Using Serpentine Structures

Due to the parallel traces that make up the serpentine, cross-coupling may occur between the individual portions of the serpentine. The cross-coupling may cause the propagation velocity and characteristic impedance of the serpentine to differ from those of a straight line of equivalent length. In general, the propagation velocity may be greater and the characteristic impedance less for the serpentine structure. To simplify the simulation environment for the CPU-cache-chip set design example, the added trace length was assumed to be equal to

the length of trace used to make the serpentine. See Figure 29.

Experiments were performed to confirm that this assumption was valid. The experiments involved Time Domain Reflectometry (TDR) and Time Domain Transmission (TDT) measurements on various serpentine configurations. The height of the serpentine, h , and the separation, s , were varied.

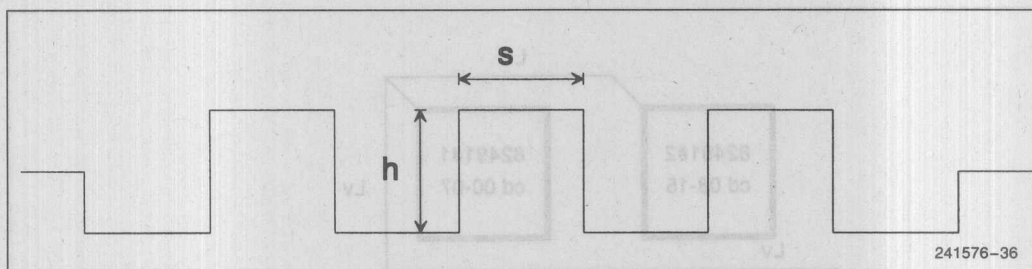


Figure 29. Parameters of a Serpentine Structure

It was found that as the separation was increased or the height decreased the propagation velocity increased. Amount of increase varied from being almost negligible to being approximately 40% for short, closely spaced serpentes. Also, it was observed that as the height decreases the magnitude of the decrease in impedance gets smaller, with the largest decrease in impedance, approximately 12%, seen when the height and separation are at their smallest. The serpentes used in the CPU-cache-chip set design example were not the worst case configuration. Based on the experiments, the serpentes should cause less than 10% decrease in the characteristic impedance and less than 30% decrease in the propagation delay.

Both of these variations appear considerable at first glance. However, serpentes, as used in the CPU-cache-chip set design example, account for only a small percentage of the entire trace length of a net. For example, if the serpentine is only 25% of the total trace length and the total propagation delay is 2 ns, representing the trace as a straight line length only introduces a maximum error of about 150 ps. This is determined by assuming the 25% of trace accounts for 0.5 ns delay and a 30% decrease is 150 ps. Based on the small amount of error introduced, it was decided that a straight line representation was accurate enough and simplified the simulations.

Note the variation in effects caused by the serpentes. Each design should perform similar analysis if a different serpentine structure than that used in the CPU-cache-chip set design example is used.

If the designer chooses to include the propagation velocity and characteristic impedance variations in the simulations, the only change is to represent the serpentine length of trace as a separate length with the different characteristics.

6.0 EXAMPLE: DESIGNING THE A12 NET FOR THE CPU-CACHE CHIP SET

The A12 net is one of the more complex nets in the optimized interface of the CPU-Cache Chip Set. The signal is driven by the Pentium processor to the 82496 cache controller and all the 82491 cache SRAMs during memory reads. In addition, the 82496 cache controller drives this signal to the Pentium processor during inquire cycles.

In routing A12 net all of the guidelines and techniques described were used. An initial routing of the A12 net was made using an H-type routing and attempting to make all of the interconnects as short as possible. The resulting topology is shown in Figure 30.

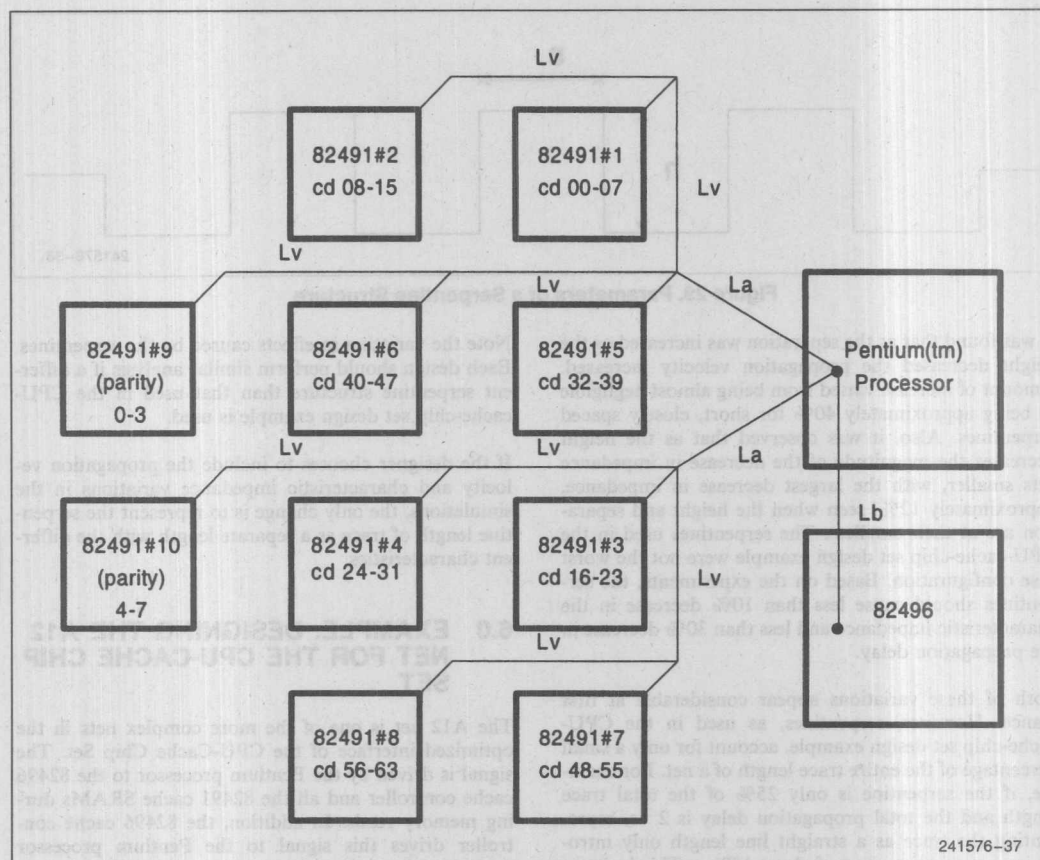


Figure 30. Initial Topology for A12 Net

The A12 net was simulated assuming the Pentium processor is driving the net. Quad Design's TLC was used to simulate the A12 net. For flight time the slow corner is used. The slow corner uses the model parameters as defined in Table 5. The actual values can be obtained from the *Pentium™ Processor User's Manual*. A TLC control file calls the appropriate model and topology files along with setting the needed measurement points to complete the flight time simulations.

Initially, the line lengths or segments between components was assumed to be the straight line distance. In other words, the initial routing conserved space and used the shortest line possible to connect the components. The rising and falling waveforms resulting from the TLC simulations of this routing are shown in Figure 31.

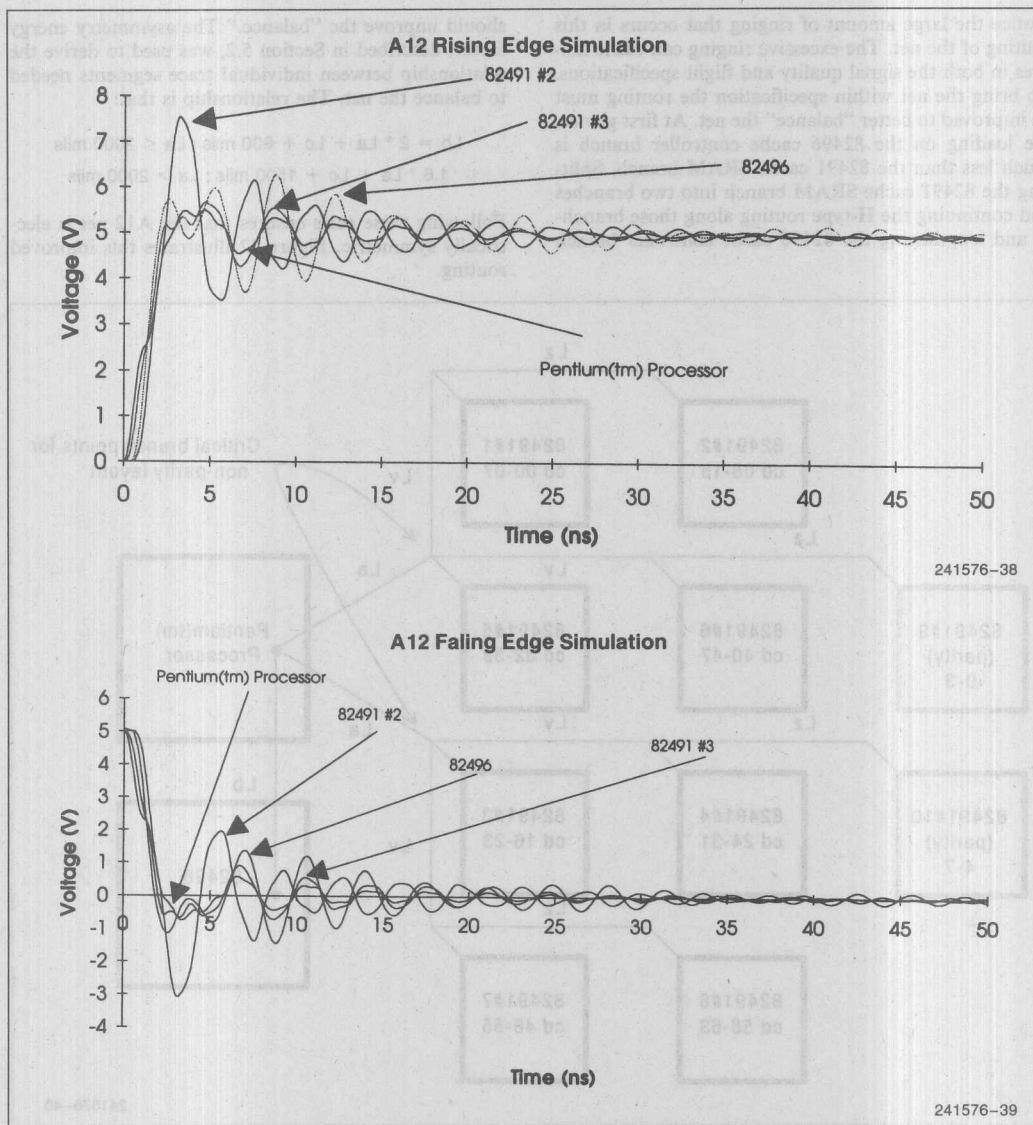


Figure 31. TLC Simulation of A12 Net Using Straight Line Lengths

same light time one must first determine the 50% point of the unloaded Pentium processor driver as shown in Figure 32. In the example this occurs at 2.25 ns. Next, one must determine where the waveform crosses the 50% Vcc point at the receiver as shown in Figure 33. This crossing occurs at 4.75 ns. The time difference between these two points is the 50-50 light time. The 50-50 light time for the A12 net example is 2.5 ns.

The improved net was also simulated using TLC and the resulting waveforms are shown in Figure 33. Notice the reduction in ringing through Figure 33. Notice the reduction in ringing. With the "balanced", or electrically symmetric routing, the net exhibits better light time and signal quality. In fact, these parameters are now within specification as summarized in Table 1.

The technique for measuring light time and signal quality are detailed in detail in Section 3.0. To mea-

Notice the large amount of ringing that occurs in this routing of the net. The excessive ringing can cause failures in both the signal quality and flight specifications. To bring the net within specification the routing must be improved to better "balance" the net. At first glance, the loading on the 82496 cache controller branch is much less than the 82491 cache SRAM branch. Splitting the 82491 cache SRAM branch into two branches and continuing the H-type routing along those branches and lengthening the 82496 cache controller branch

should improve the "balance." The asymmetry energy factor, described in Section 5.2, was used to derive the relationship between individual trace segments needed to balance the net. The relationship is that:

$$L_b = 2 * L_a + L_c + 800 \text{ mils ; } L_a \leq 2000 \text{ mils}$$

$$1.6 * L_a + L_c + 1500 \text{ mils ; } L_a > 2000 \text{ mils}$$

Following these rules ensures that the A12 net is electrically symmetric. Figure 32 illustrates this improved routing.

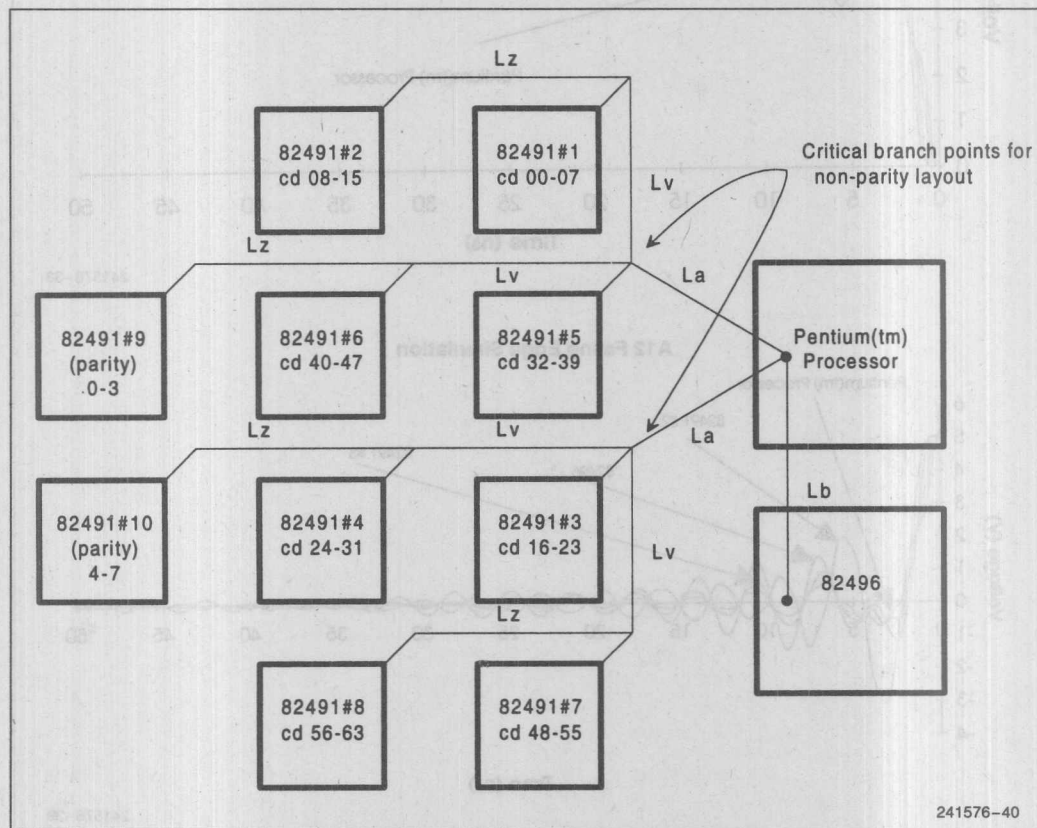


Figure 32. Improved Topology of A12 Net

The improved net was also simulated using TLC and the resulting waveforms are shown in Figure 33 through Figure 35. Notice the reduction in ringing. With the "balanced" or electrically symmetric routing the net exhibits better flight time and signal quality. In fact these parameters are now within specification as summarized in Table 7.

The technique for measuring flight time and signal quality are described in detail in Section 2.0. To mea-

sure flight time one must first determine the 50% point of the unloaded Pentium processor driver as shown in Figure 33. In the example this occurs at 2.26 ns. Next one must determine where the waveform crosses the 50% Vcc point at the receiver as shown in Figure 34. This crossing occurs at 4.54 ns. The time difference between these two points is the 50-50 flight time. The 50-50 flight time for the A12 net example is 2.28 ns.

Flight time also assumes the waveform continues through the 50% Vcc point with a slope of at least 1 V/ns through the 65% Vcc point. To ensure this, first determine the 65% point on the A12 flight time simulation as shown in Figure 35. The 65% Vcc point is 5.32 ns. Next extrapolate using the 1V/ns line to find where it crosses the 50% voltage level by subtracting 0.68 ns from the 65% Vcc number. The extrapolated

50% Vcc point for the example is 4.64 ns. The time difference between the unloaded buffer's 50% point and the extrapolated crossing of the 50% point is the 50–65 flight time. The 50–65 flight time is 2.38 ns.

The greater of the 50–50 and 50–65 flight times is the flight time for the net. In this case, the flight time is 2.38 ns, the 50–65 flight time.

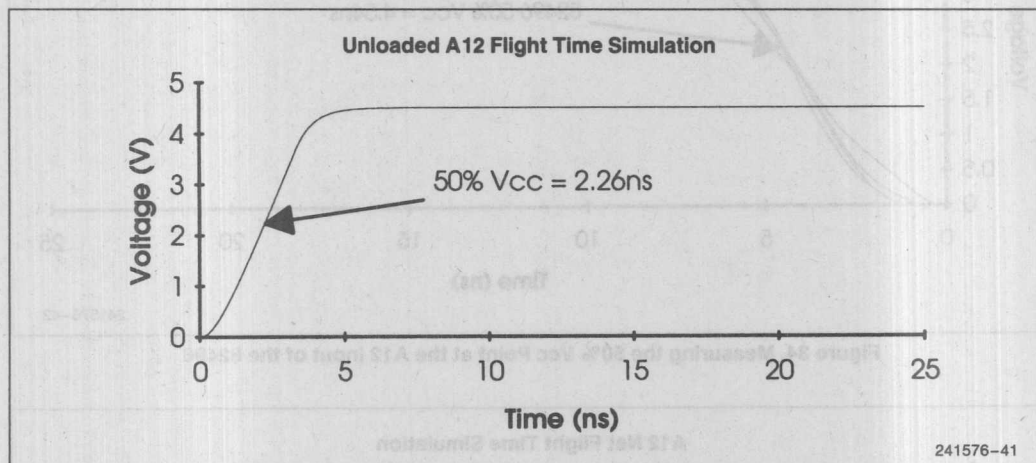


Figure 33. Measuring the 50% Vcc Point of the Unloaded Output

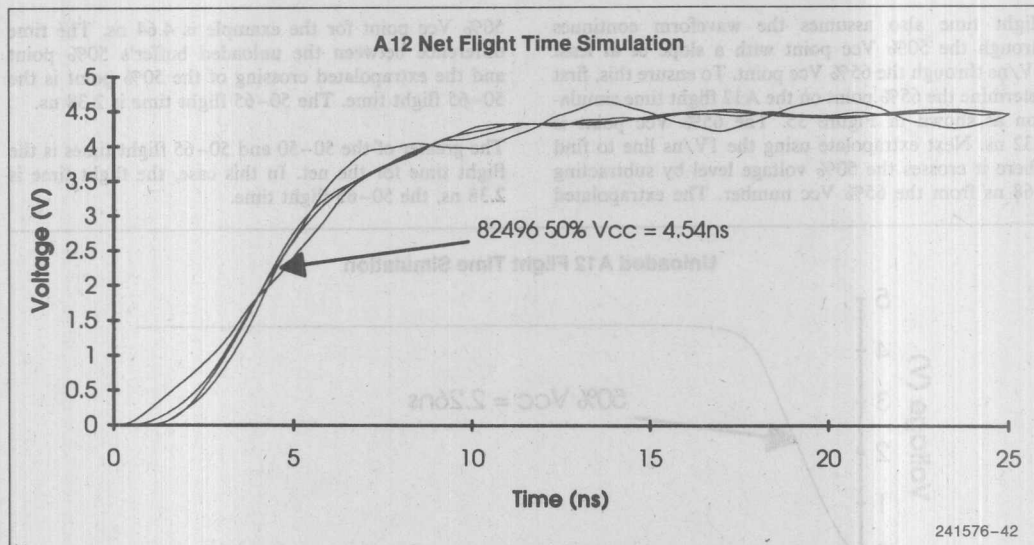


Figure 34. Measuring the 50% Vcc Point at the A12 Input of the 82496

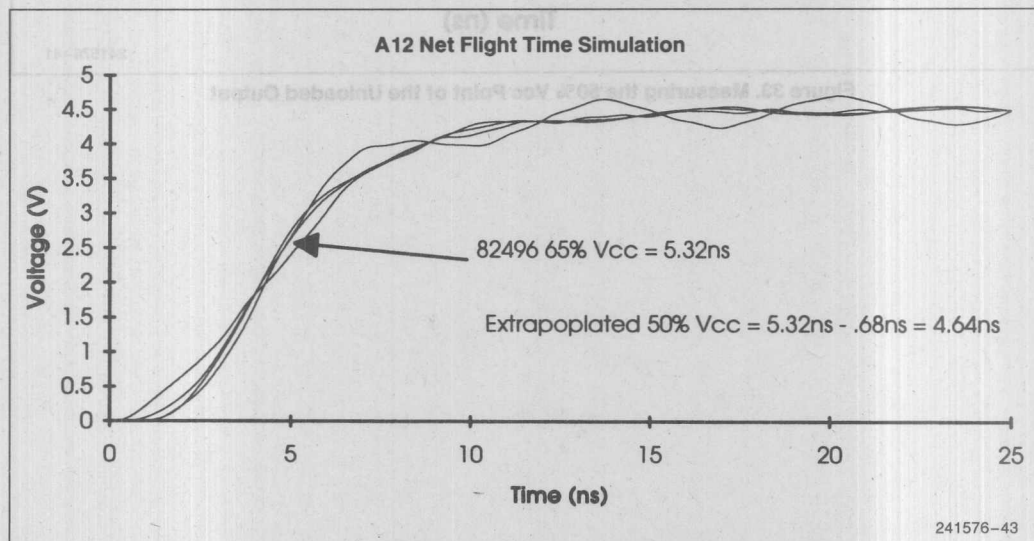


Figure 35. Measuring the 65% Vcc Point at the A12 Input of the 82496

nal quality parameters for the A12 net. Overshoot is the maximum voltage above Vcc. Time beyond supply is

maximum voltage amount that the signal cross back across Vcc. Settling time is measured from point C and D.

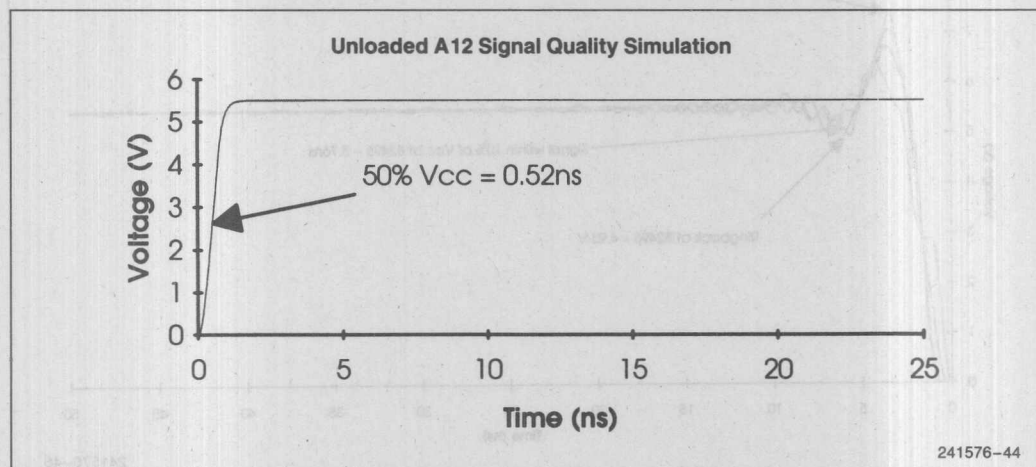


Figure 36. Measuring the 50% Vcc Point of the Unloaded Output at the Fast Corner

3

Table 7. Flight Time and Signal Quality Simulated Values

Signal	Flight Time		Overshoot		Ringback		Settling Time
	Spec	TLC	Spec	TLC	Spec	TLC	
A12	28 ns	23.8 ns	3.0V	1.88V	35% Vcc	0.67V	12.5 ns
							5.76 ns
							TLC

available to the components and the board. Intel plans to update this example accordingly.

The intent of the design example is to provide system designers a starting point. It provides one solution to how the Pentium processor, 8255 cache controller, and 82421 cache SRAM components can be placed and routed to ensure flight time and signal quality specifications are met. It is not the only solution. System designers can alter the layout to meet their system requirements as long as the flight time and signal quality specifications are met.

7.0 285K CPU-CACHE CHIP SET OPTIMIZED INTERFACE LAYOUT DESIGN EXAMPLE

This chapter contains an example layout design for Intel's 285K CPU-Cache Chip Set's optimized interface. Intel has simulated and verified the example layout using the latest information. Work is currently underway to validate the design by measuring the flight time and signal quality parameters on boards based on the design example. As updated information becomes

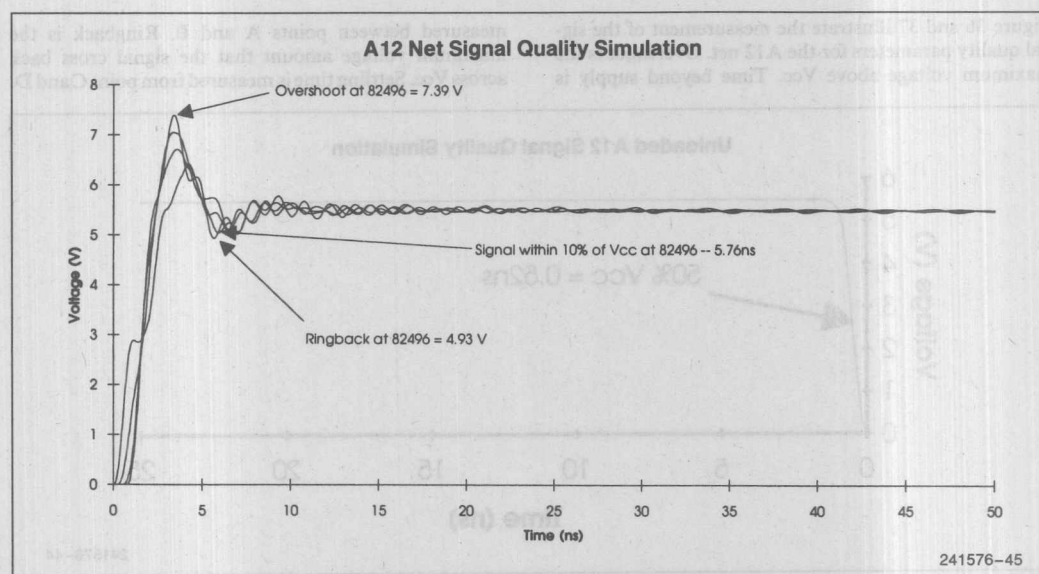


Figure 37. Measuring the Signal Quality of the 82496

The flight time and signal quality specifications for this net are listed in Table 7 along with the values measured in the simulation of the net.

Table 7. Flight Time and Signal Quality Simulated Values

Signal	Flight Time		Overshoot		Ringback		Settling Time	
	Spec	TLC	Spec	TLC	Spec	TLC	Spec	TLC
A12	28 ns	2.38 ns	3.0V	1.89V	35% Vcc	0.57V	12.5 ns	5.76 ns

7.0 256K CPU-CACHE CHIP SET OPTIMIZED INTERFACE LAYOUT DESIGN EXAMPLE

This chapter contains an example layout design for Intel's 256 Kbyte CPU-Cache Chip Set's optimized interface. Intel has simulated and verified the example layout using the latest information. Work is currently underway to validate the design by measuring the flight time and signal quality parameters on boards based on the design example. As updated information becomes

available on the components and the boards, Intel plans to update this example accordingly.

The intent of the design example is to provide system designers a starting point. It provides one solution of how the Pentium processor, 82496 cache controller, and 82491 cache SRAM components can be placed and routed to ensure flight time and signal quality specifications are met. It is not the only solution. System designers can alter the layout to meet their system requirements as long as the flight time and signal quality specifications are met.

7.1 Layout Objectives

The 256K layout is an example of a CPU-Cache chip set arrangement that meets Intel's chip set specifications. The layout consists of 1 Pentium processor, 1 82496 cache controller, and 10 82491 cache SRAMs for a 256K second-level cache with parity. Although the layout is specifically designed for a chip set with parity, we will also discuss conversion to a non-parity layout.

This example layout follows the chip set's flight time and signal quality specifications. In addition to meeting those specifications, we had the following objectives:

1. To design the optimized portion of the interface so that the layout is not limited by interconnect performance. By not artificially creating any critical paths, the interface can yield maximum performance of the chip set.
2. To be consistent with EMI and thermal requirements.
3. To have the layout be used as a validation and correction vehicle by Intel. Intel will use the layout to validate the optimized interface of the chip set, mea-

sure flight times and signal quality, and tune input and output buffers.

Provided are complete specifications for a board layout: part lists, board layer plots, and the electronic files in Gerber format. Also provided are a set of topologies and line lengths so it will be easy to understand how the layout was generated.

7.2 Component Placement

To meet flight time with clock skew restrictions we placed the parts in relative proximity to each other. At the same time, we ensured that the layout's Memory Bus Controller (MBC) interface signals are routable. Figure 38 illustrates how the chip set components are placed in the layout example. The dot indicates the location of pin 1. Figure 38 component side view of the layout.

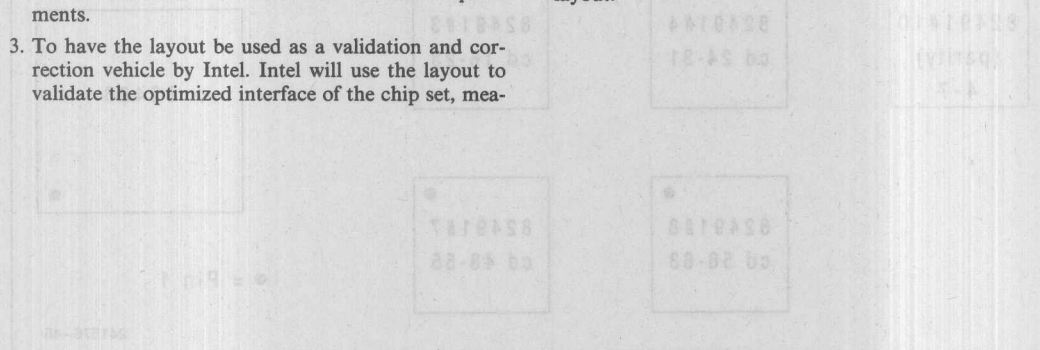


Figure 38. Component Placement

for routing the "point-to-point" signals such as CADs.

Topologies are also supplied for the external interfaces. These topologies provide standards for routing signals from the chip set components to the peripheral where they can be connected to the memory bus and memory bus controller (MBC). However, topologies are not supplied for point-to-point signals in the MBC interface (e.g., CRDY#). Instead, the system designer must optimize those for the particular application.

Table 9 lists the topologies provided for the MBC interface signals which are not point-to-point.

7.3 Signal Routing/Topologies

Tables 8 and 9 list the signal nets and their corresponding topologies for the optimized and external interfaces of the CPU-Cache Chip Set.

All chip set signals in the optimized interface fall into six groups: bus addresses, high address, bus data, and bus control. 32-bit control, CPU data, and bus control. Within each group are subsets of signals that share common termination and destination points. Each subset has a unique routing called a "topology". Groups, signals, and topologies are listed in Table 8.

Topologies are given only for signals that are routed multiple times. It is the system designer's responsibility

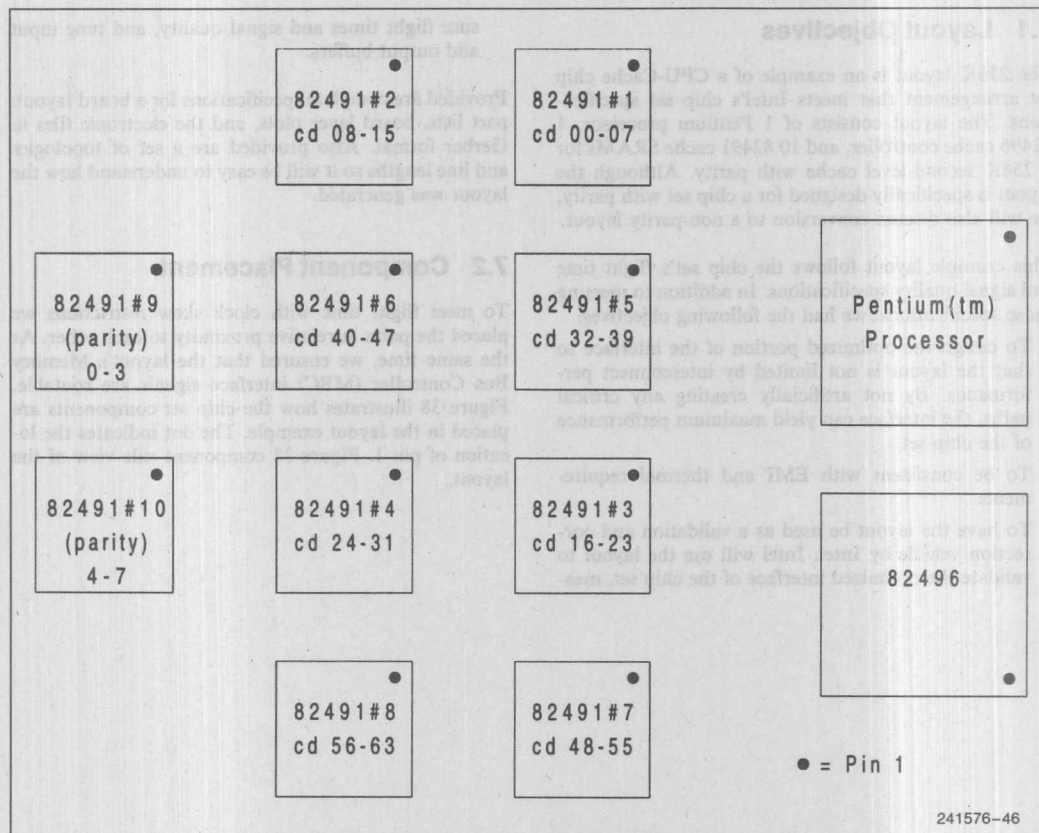


Figure 38. Component Placement

7.3 Signal Routing/Topologies

Tables 8 and 9 list the signal nets and their corresponding topologies for the optimized and external interfaces of the CPU-Cache Chip Set.

All chip set signals in the optimized interface fall into six groups: low addresses, high addresses, Pentium processor control, 82496 control, CPU data, and byte enables. Within each group are subsets of signals that share common origination and destination points. Each subset has a unique routing called a "topology." Groups, subsets, and topologies are listed in Table 8.

Topologies are given only for signals that are routed to multiple chips. It is the system designer's responsibility

for routing the "point-to-point" signals such as CADs#.

Topologies are also supplied for the external interface. These topologies provide channels for routing signals from the chip set components to the periphery where they can be connected to the memory bus and memory bus controller (MBC). However, topologies are not supplied for point to point signals in the MBC interface (e.g. CRDY #). Instead, the system designer must optimize these for the particular application.

Table 9 lists the topologies provided for the MBC interface signals which are not point to point.

Table 8. Optimized Interface Signal Net/Topology Assignments

Grouping	Routing Requirements	Topology
Low Addresses		
(PA3–PA16)	Bused to all core components. Must be routed to optimize delay and signal quality at all points.	1
High Addresses		
(PA17–PA31, PBT0–PBT3)	Point to point links. Must be kept as short as possible.	4
Pentium™ Processor Control		
(HITM#, W/R#)	Same as low addresses.	1
(ADS#)		3b
(ADSC#, AP, CACHE#, D/C#, LOCK#, M/IO#, PCD, PWT, SCYC)	Same as high addresses.	4
CC Control		
(BRDYC2#, WRARR#, MCYC#, WAY, BUS#, MAWEA#, WBWE#, WBTYP#, WBA, BLAST#)	Must be routed to optimize delay and signal quality at the CS.	3
(BLEC#)	Not routed to parity CSs.	3a
(BOFF#)		1b
(AHOLD, EADS#, KEN#, BRDYC1#, INV, EWBE#, NA#, WB/WT#)	Same as high addresses.	4
CPU Data		
(CD0–CD63)	Point to point signals. Keep as short as possible. Keep the total length of each trace within 1/2" of each other to minimize skew.	4
Byte Enables		
(CBE0#–CBE7#)		5

3

Table 9. External Interface Signal Net/Topology Assignments

Signal	Topology
RESETC50, CRDY0#	10
CRDY#, RESETC51	11
MBRDY#, MOCLK, MDOE#	12, 13
MFRZ#, MSEL#, MZBT#, MCLK	14
BRDY#, CLK0	15
CLK1, BRDY1#, MEOC1#	16
CLK2, BRDY2#, MEOC2#	17
CLK3, BRDY3#, MEOC3#	18
MDATA0–63	19

Figures 39 through Figure 58 are the topologies which are described in Tables 8 and 9. A topology is a graphical representation how specific sets of signals are routed.

A topology shows the components that share a specific signal and the relative lengths of the traces between components.

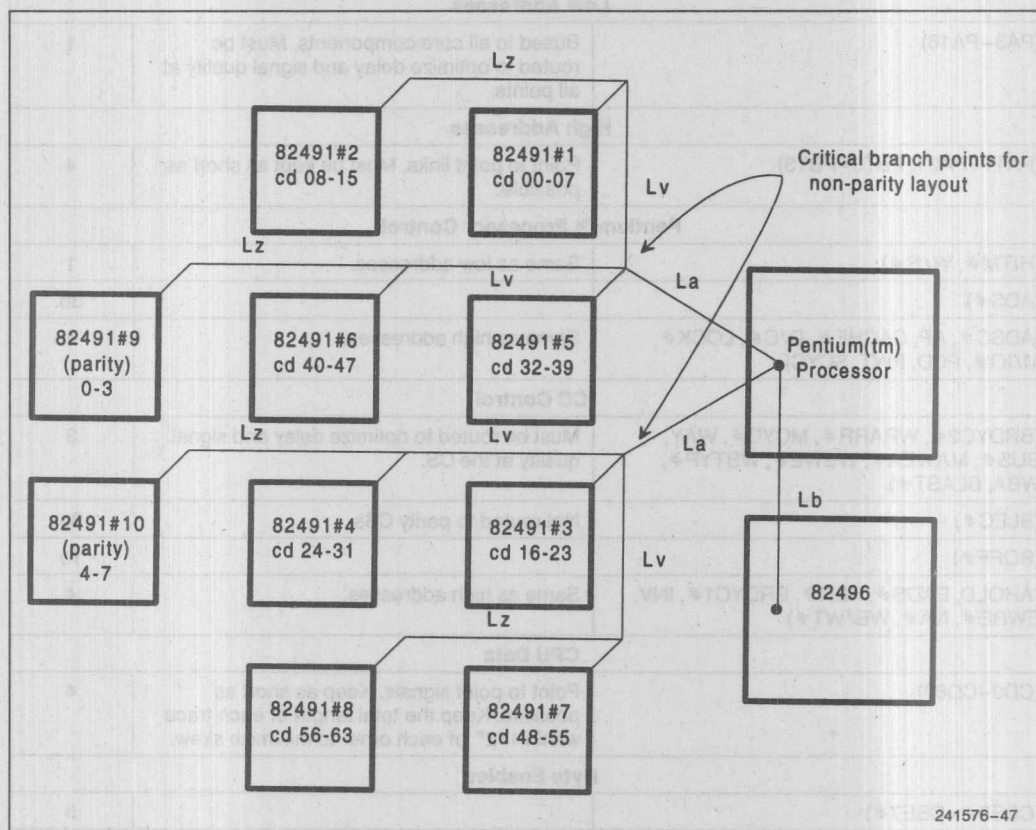


Figure 39. Topology 1

Topology	Signal
10	RESETCD, CORDY+
11	CORDY+, RESETCD
12	MEDY+, MOCIL, MDOE+
13	MERZ+, MSEL+, MEST+, MOK
14	BRDY+, CLK0
15	CLK1, BRDY1+, MEDC1+
16	CLK2, BRDY2+, MEDC2+
17	CLK3, BRDY3+, MEDC3+
18	DATA0-63

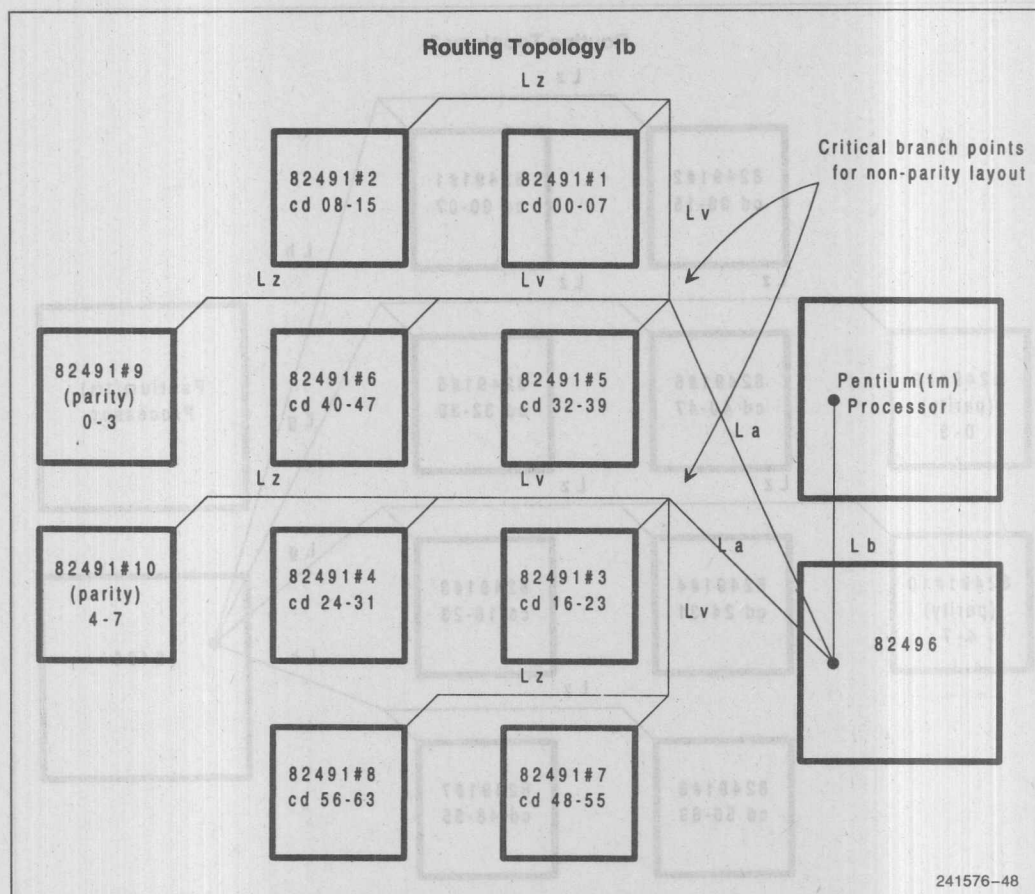


Figure 40. Topology 1b

3

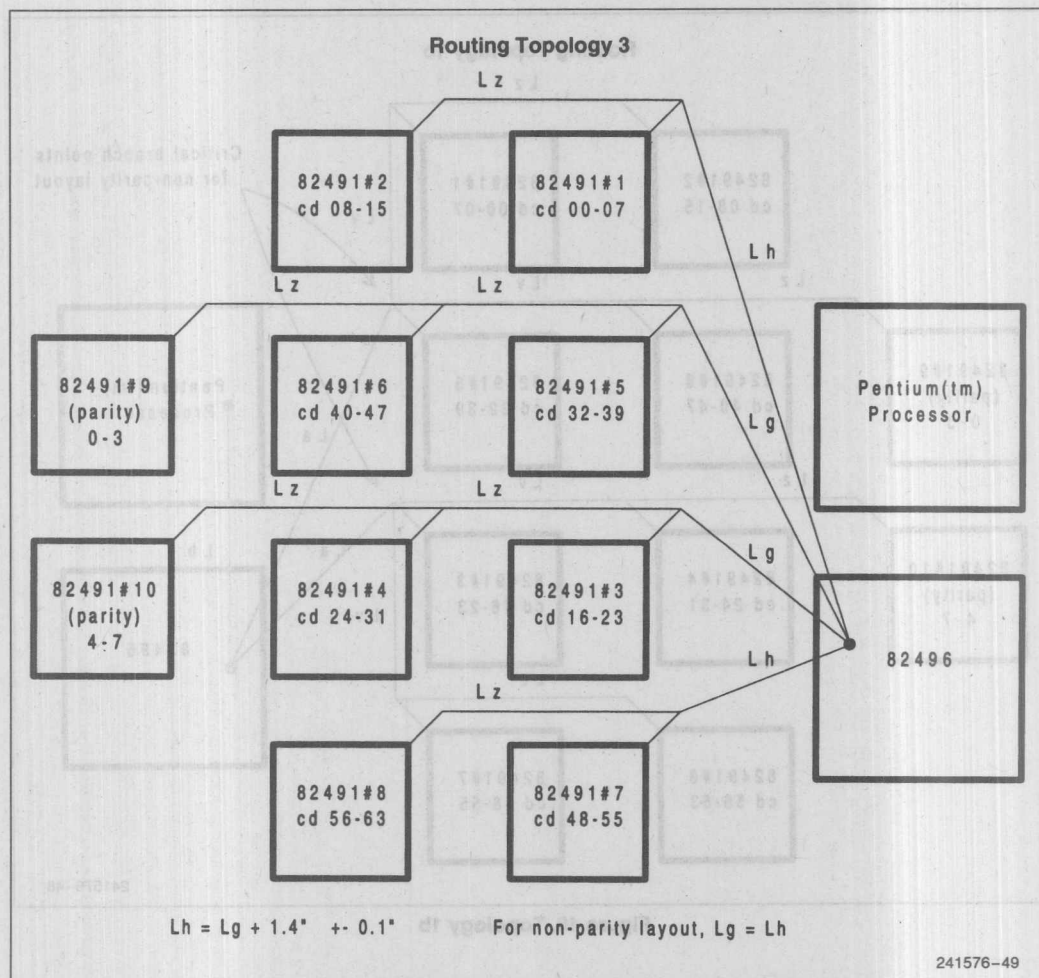


Figure 41. Topology 3

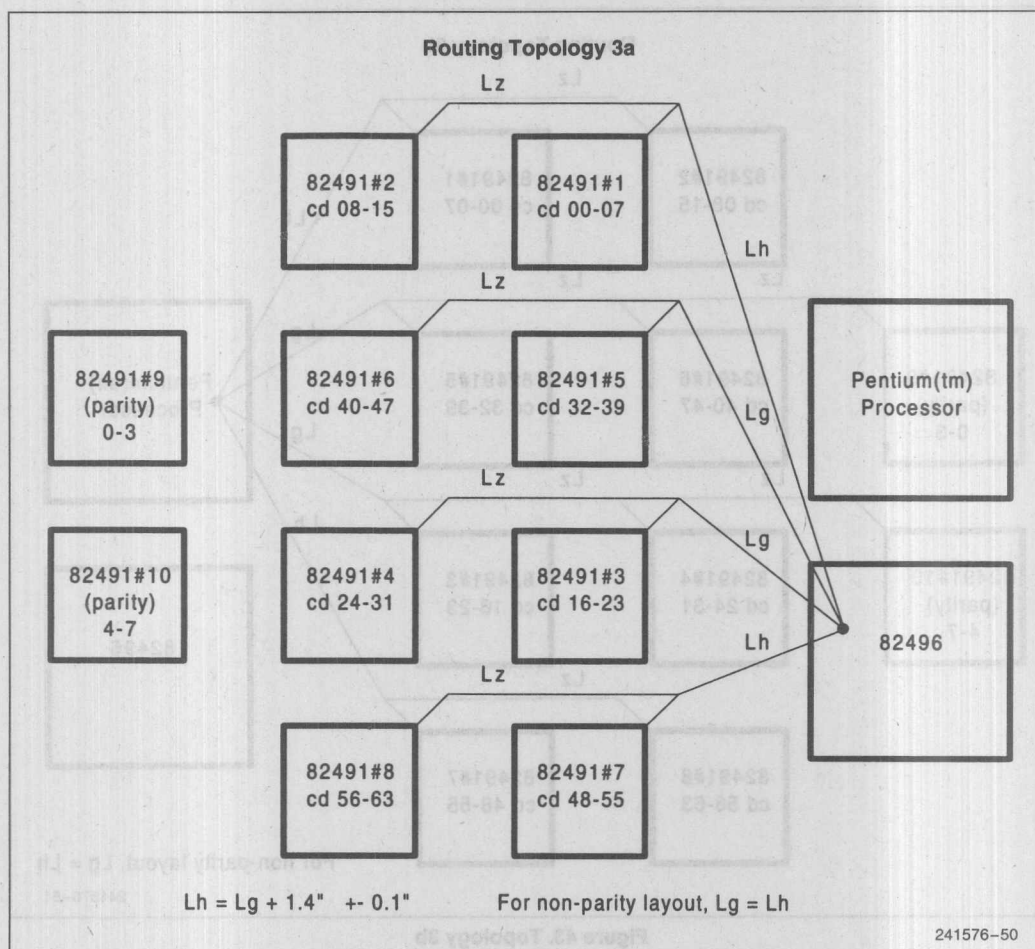


Figure 42. Topology 3a

3

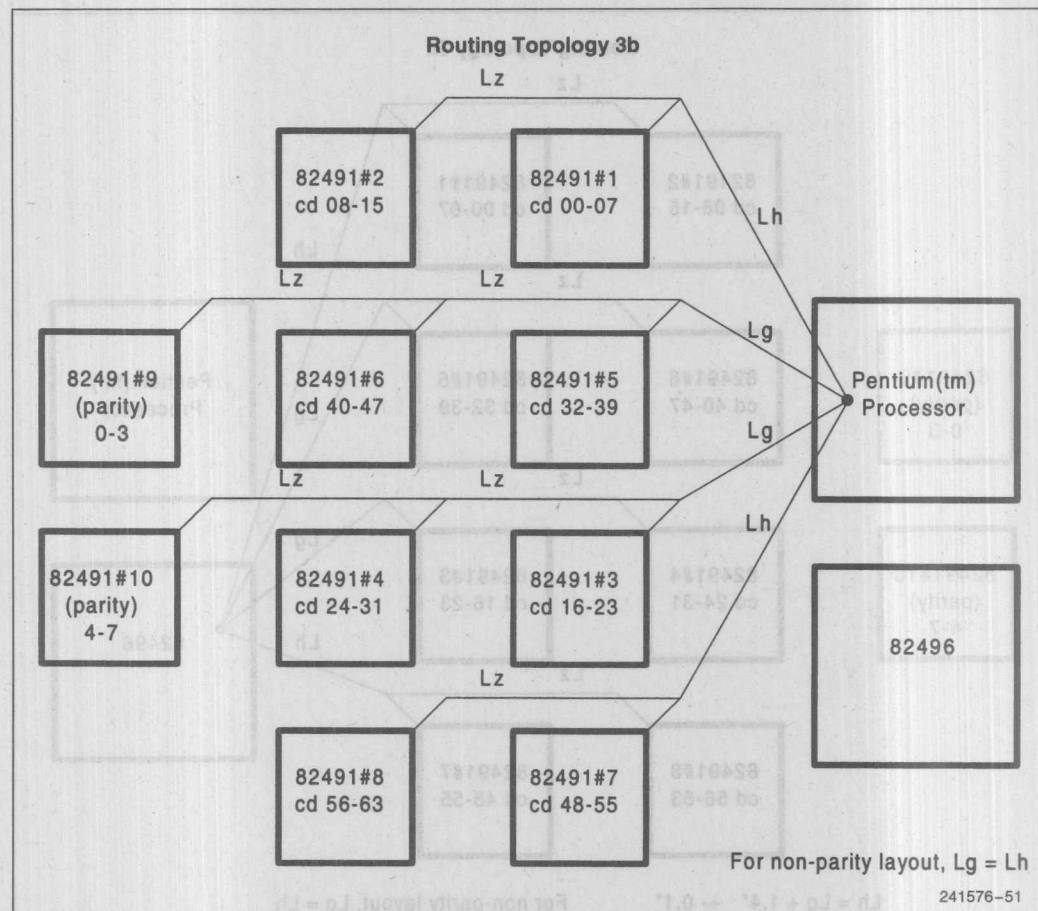


Figure 43. Topology 3b

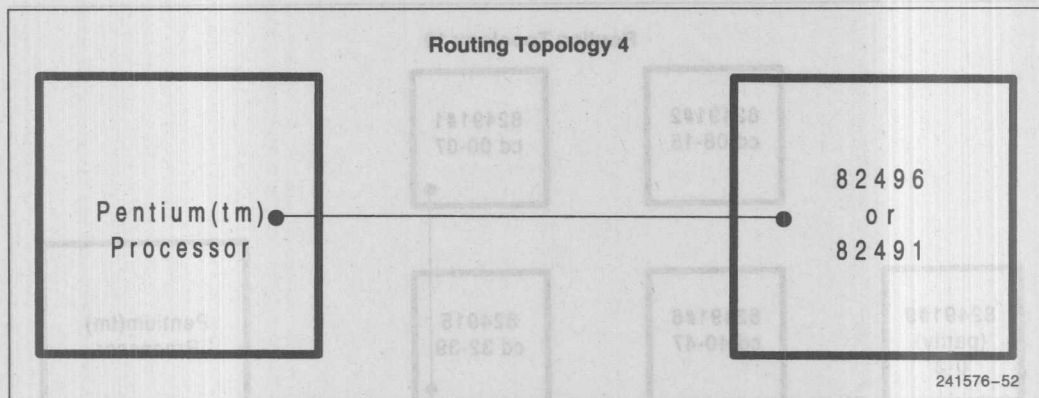


Figure 44. Topology 4

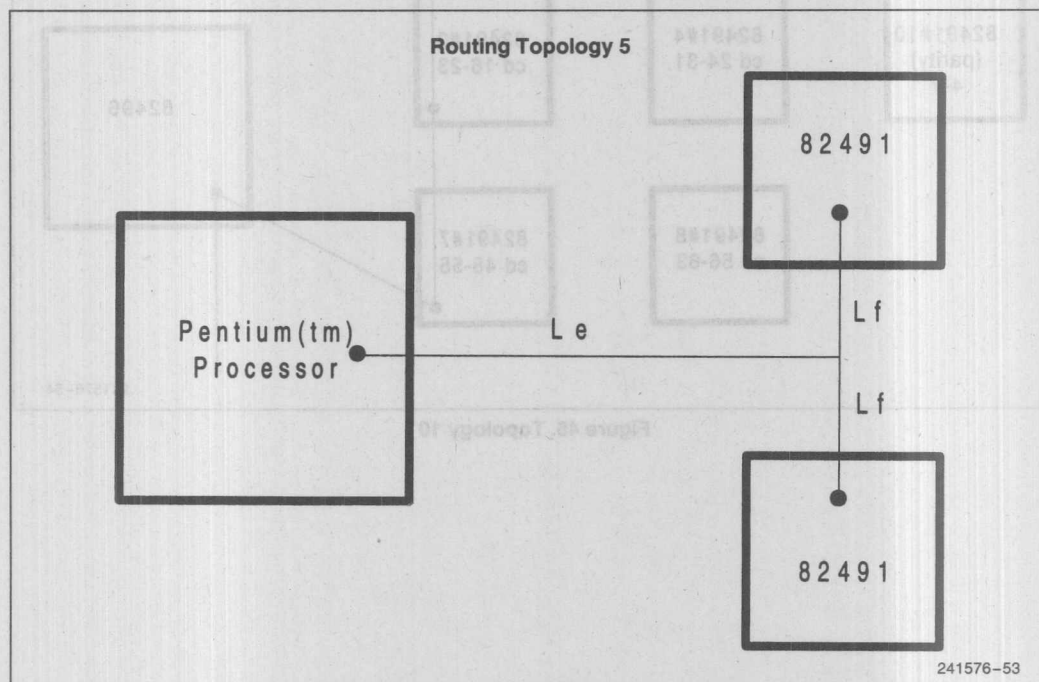


Figure 45. Topology 5

3

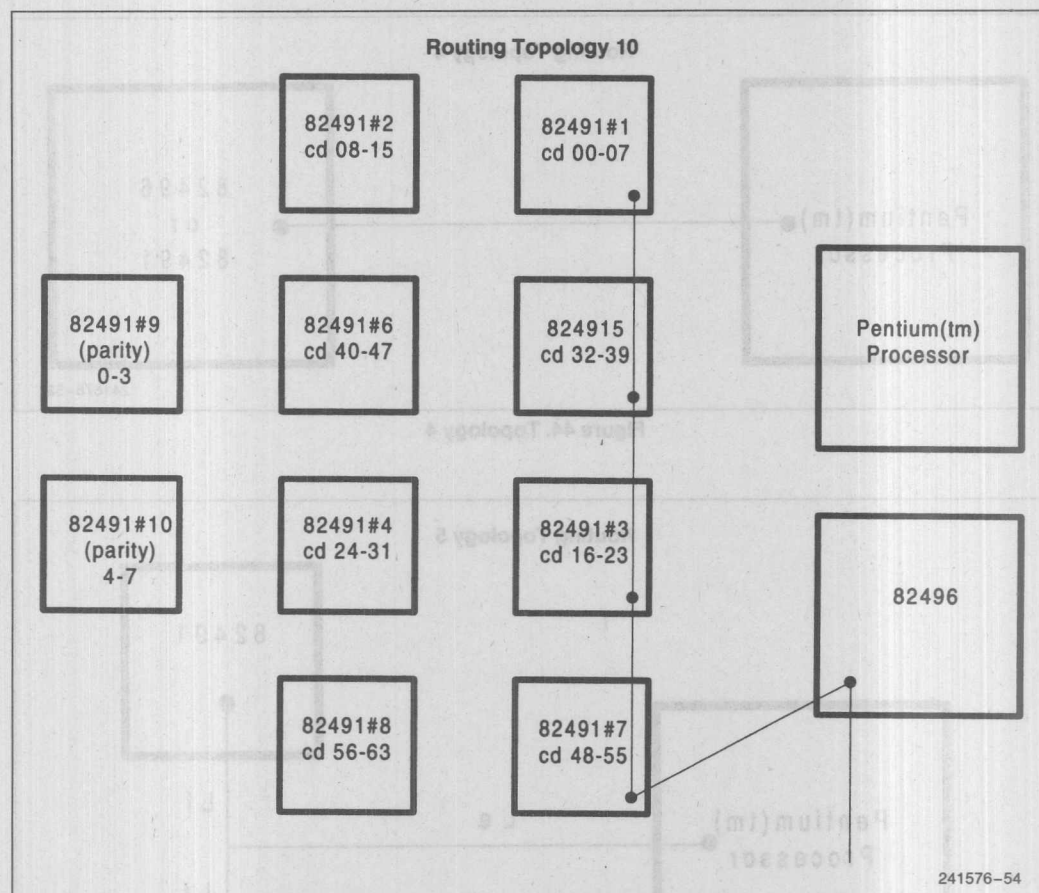


Figure 46. Topology 10

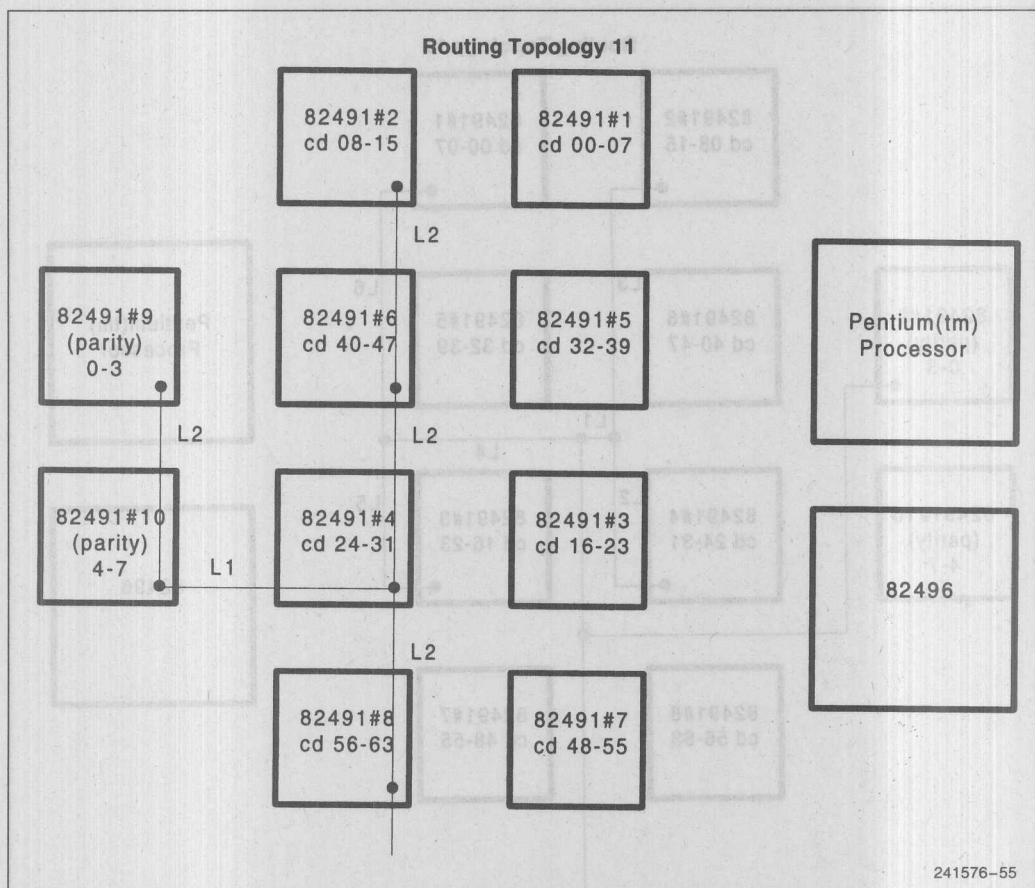


Figure 47. Topology 11

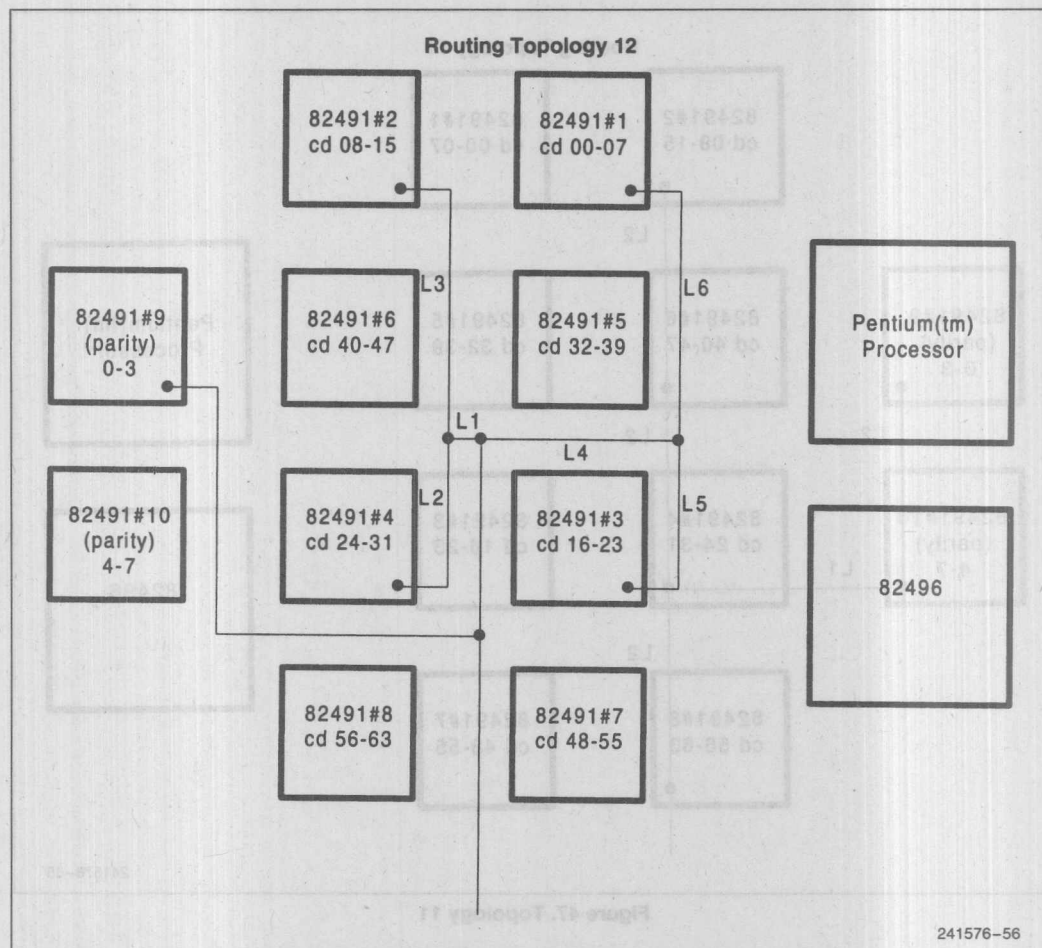


Figure 48. Topology 12

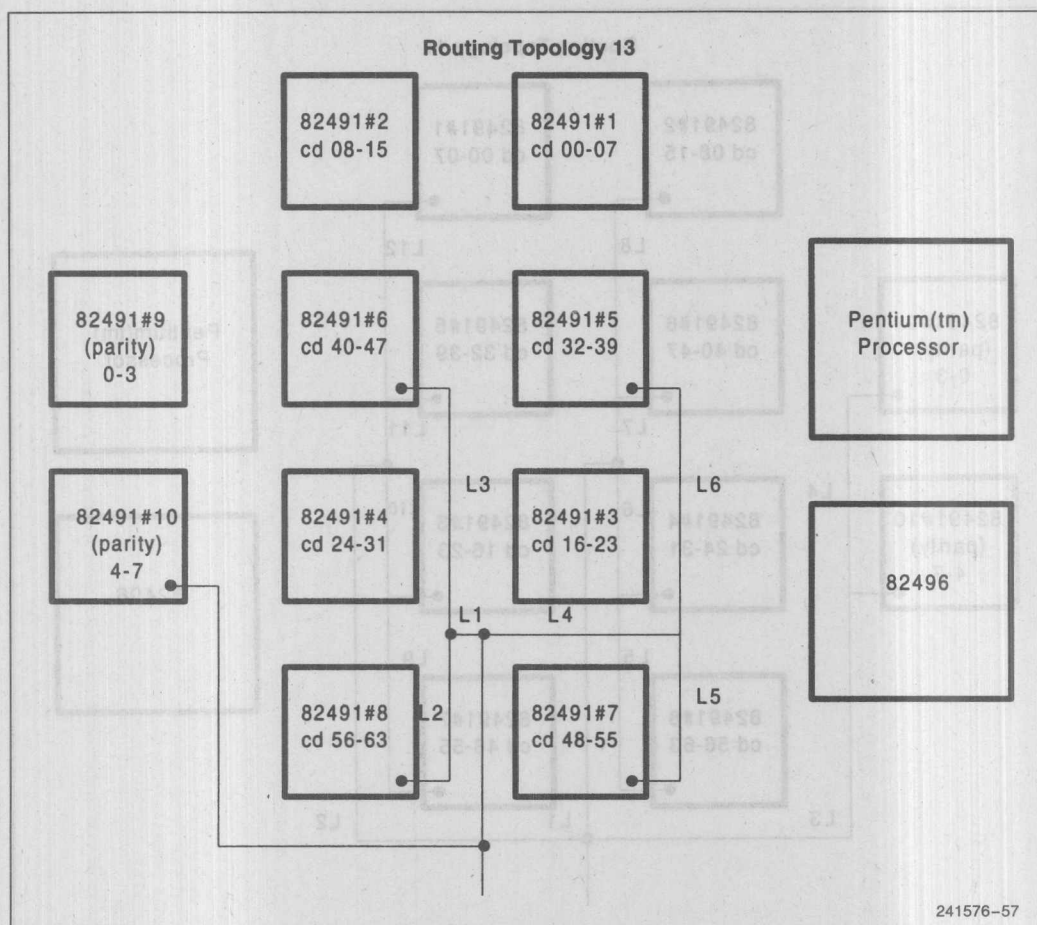


Figure 49. Topology 13

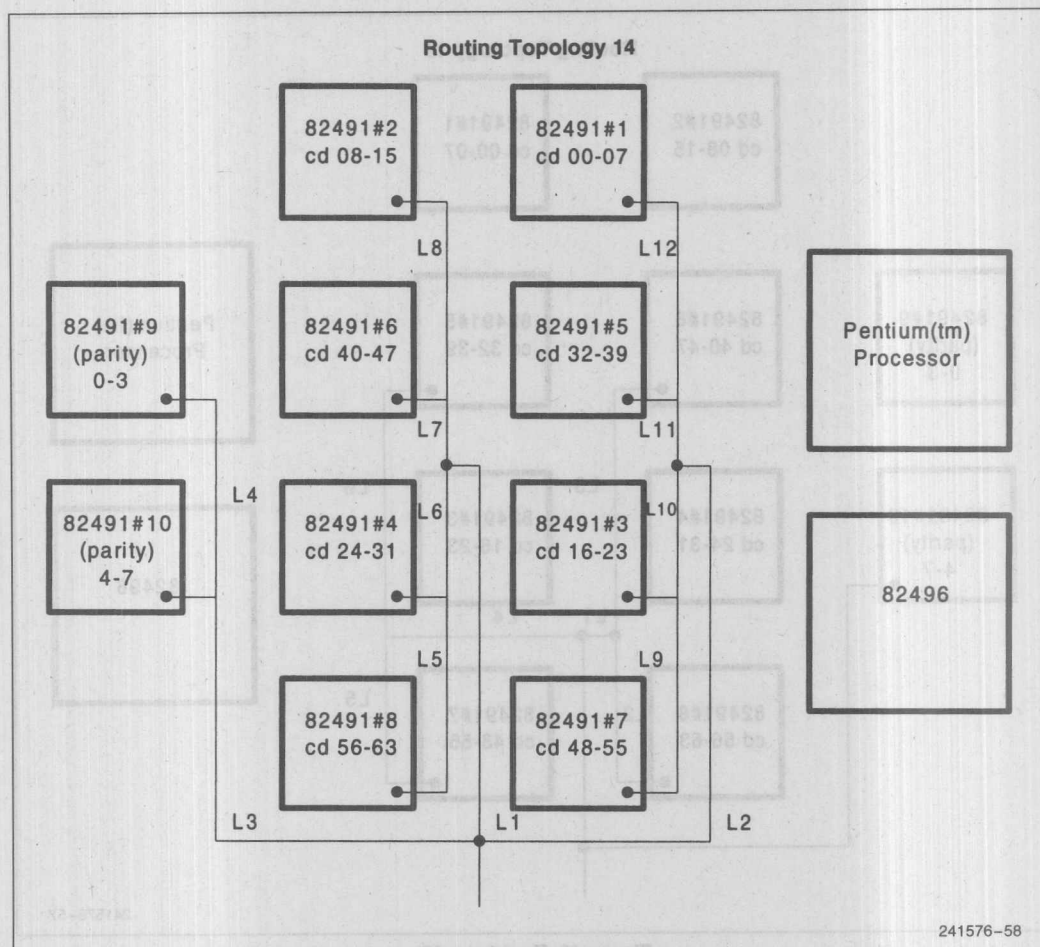


Figure 50. Topology 14

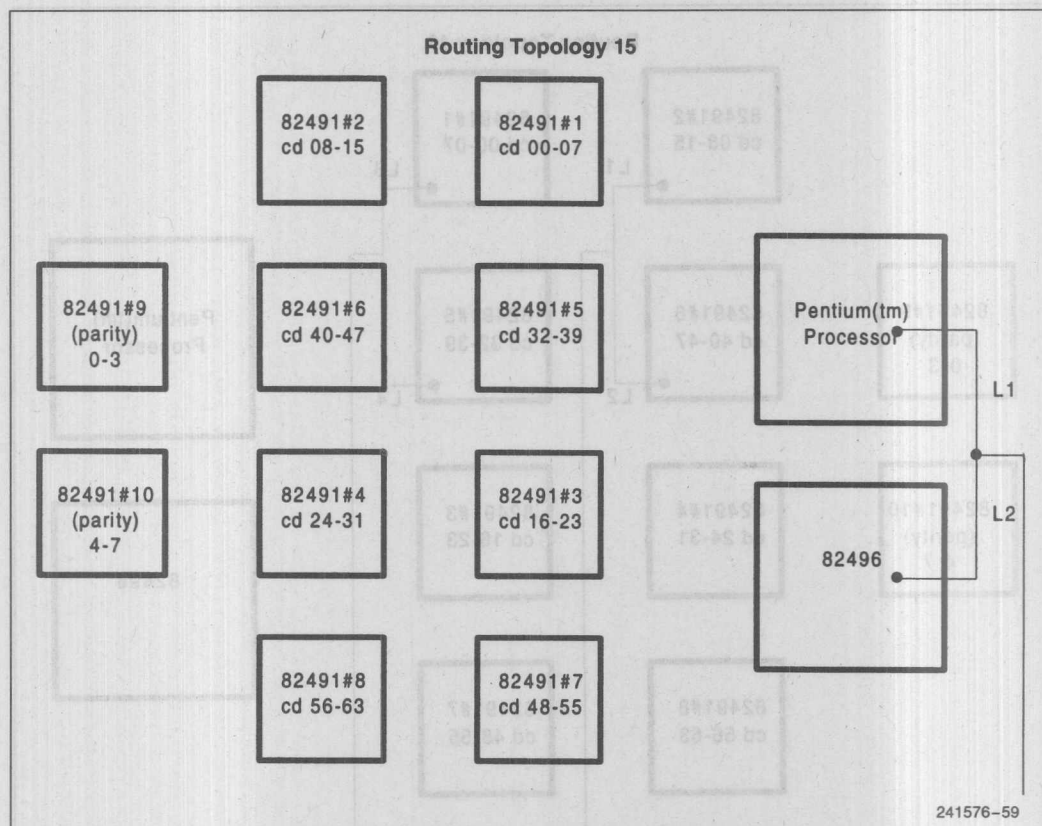


Figure 51. Topology 15

3

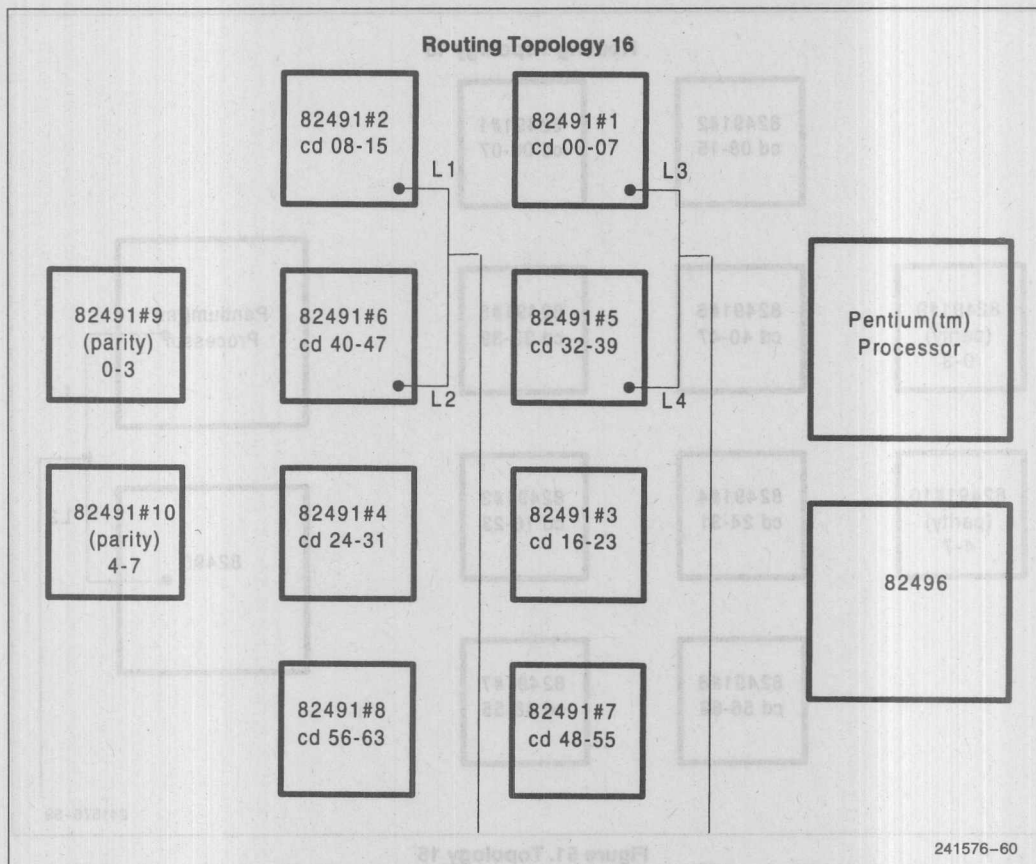


Figure 52. Topology 16

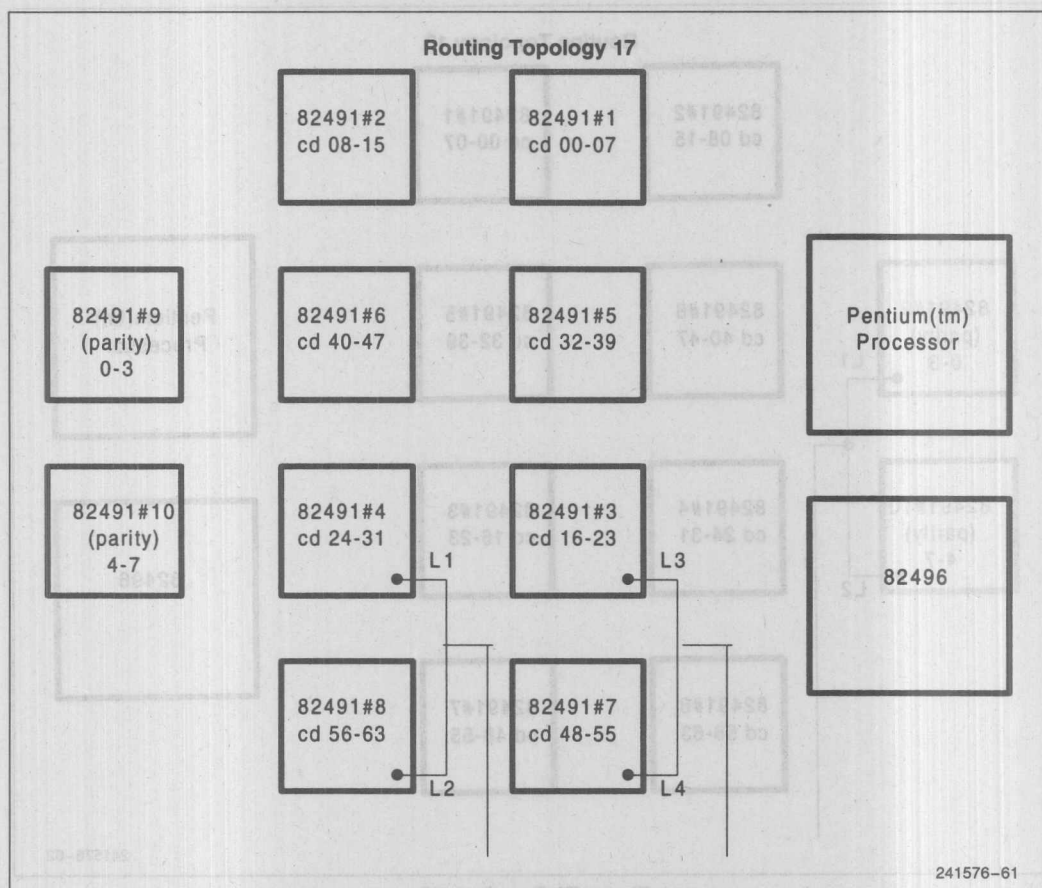


Figure 53. Topology 17

3

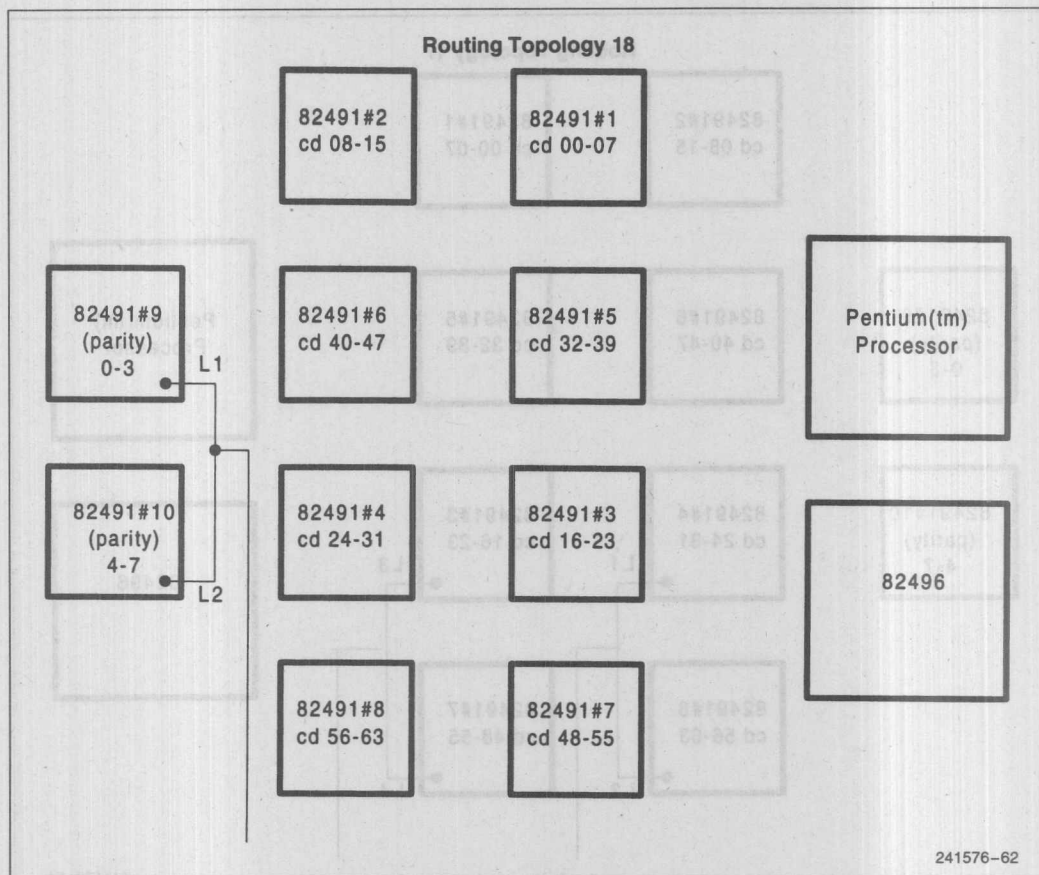


Figure 54. Topology 18

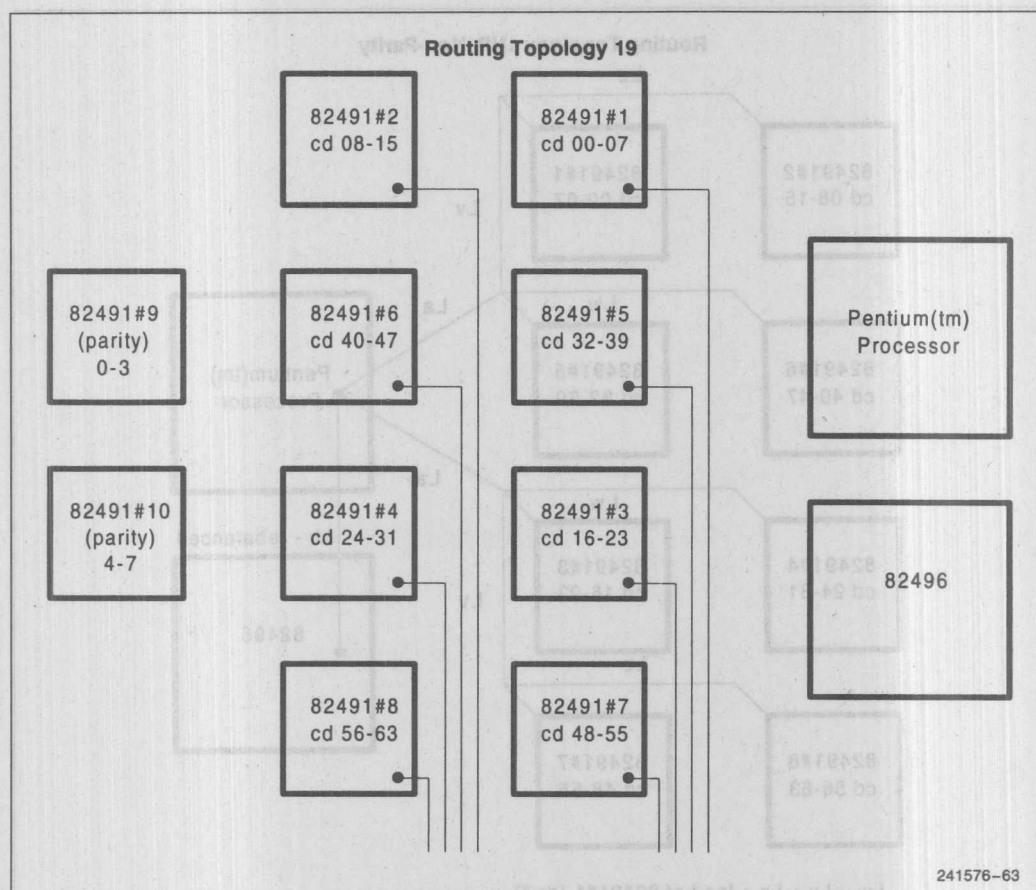


Figure 55. Topology 19

Figures 56 and 57 provide topologies for the non-parity configuration of the 256 Kbyte CPU-Cache Chip Set.

Refer to Section 7.7.1 for more details on the non-parity configuration.

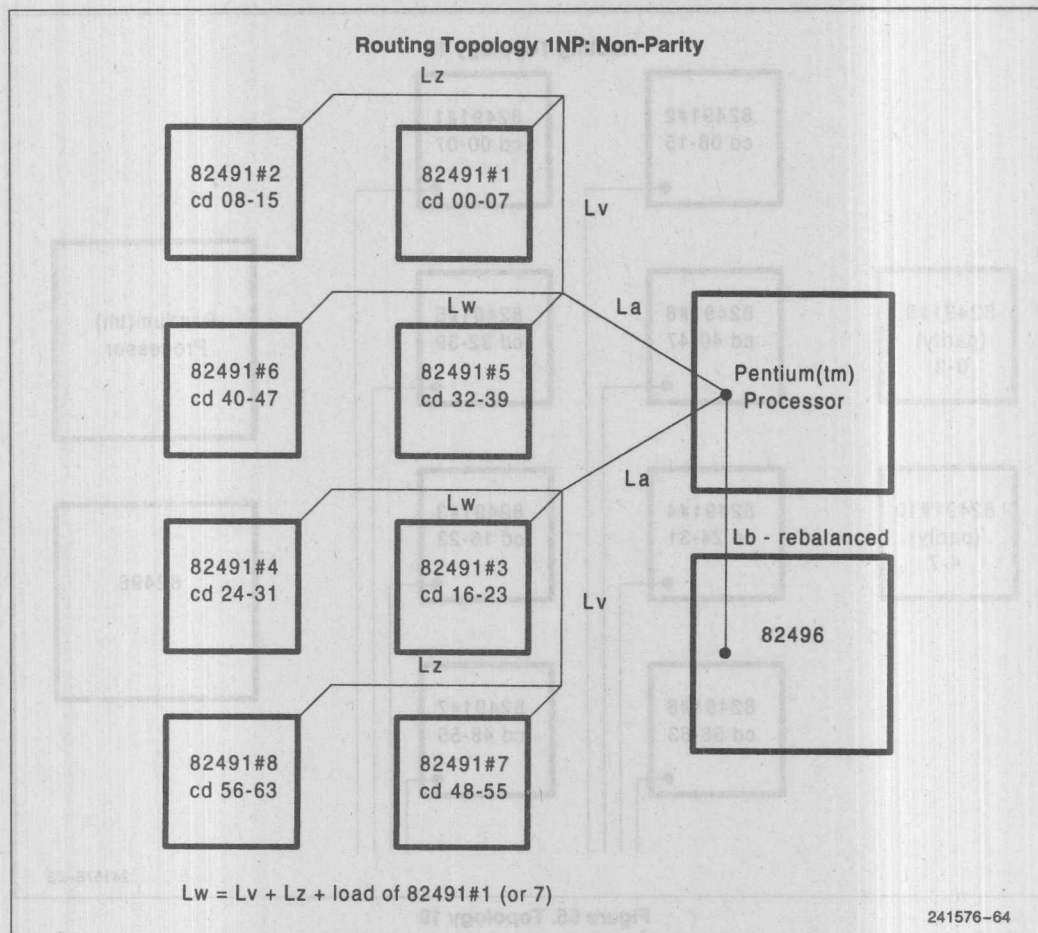


Figure 56. Topology 1NP

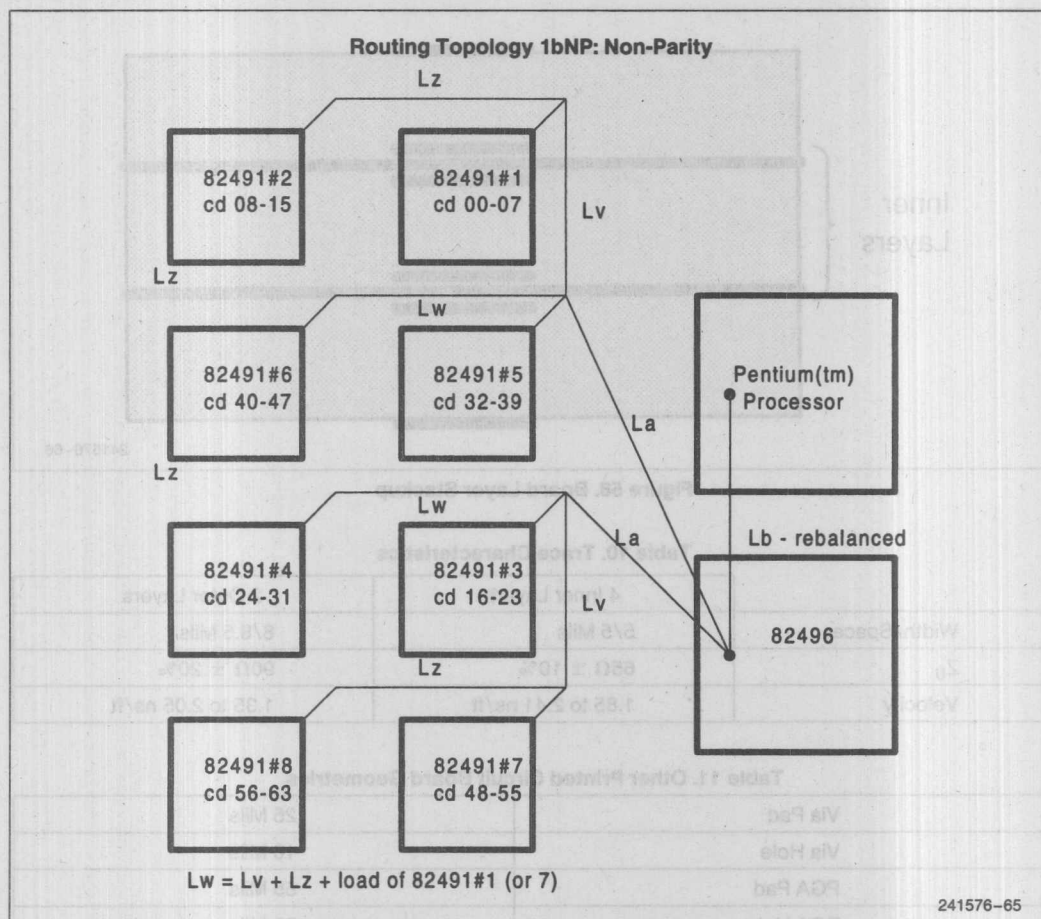


Figure 57. Topology 1bNP

7.4 Board/Trace Properties

Specific board and trace properties were assumed while performing the simulations to optimize the chip set layout. These properties were used as the specification or guideline the board manufacturer was to use in building boards. Figure 58 provides the board layer stackup.

Table 10 lists the minimum and maximum trace characteristics. These parameters along with the board material determine the spacing between layers and the total board thickness. See Table 11.

Only the inner layers of the board are impedance controlled. The top and bottom layers are not impedance controlled.

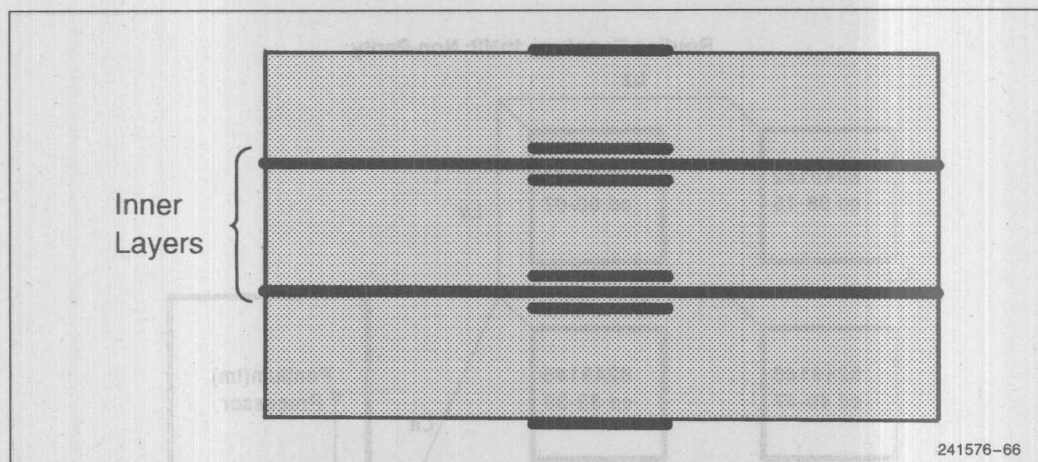


Figure 58. Board Layer Stackup

Table 10. Trace Characteristics

	4 Inner Layers	2 Outer Layers
Width/Space	5/5 Mils	8/8.5 Mils
Z_0	$65\Omega \pm 10\%$	$90\Omega \pm 20\%$
Velocity	1.85 to 2.41 ns/ft	1.35 to 2.05 ns/ft

Table 11. Other Printed Circuit Board Geometries

Via Pad	25 Mils
Via Hole	10 Mils
PGA Pad	55 Mils
PGA Hole	38 Mils
Layout Grid	5 Mils

7.5 Design Notes

The following design notes accompany this layout example:

1. The layout did not specifically address heat dissipation except to allow space for heat sinks to be attached. Please see the *Pentium™ Processor User's Manual* for the devices' thermal specifications. The *Pentium Processor Thermal Design Guidelines* application note provides some examples of possible thermal solutions.
2. All fast-switching signals are routed near the power and ground planes on inner layers of the board to minimize EMI effects. However, two sets of signals are routed on the top layer of the board: BRDYC1#, and JTAG signals. BRDYC1# is routed on top to take advantage of the higher trace velocity there. JTAG signals are routed on the top layer because they are low-speed signals and will probably be re-routed by each customer to suit individual needs.

3. Resistor R1 (0) is used to set the Pentium processor configurable output buffers (A3-A20, ADS#, W/R#, and HITM#). When the resistor is included the buffers are set to the Extra Large size. When it is not included (BUSCHK# internally pulled high) the buffers are set to Large size. Intel currently recommends the large buffers be used for the 256K layout example. The 0 Ω resistor should be designed into your design as Intel may change the recommended buffer size once silicon and the system design have been characterized.
4. The 82496 output buffers that drive the 82491 inputs must also be configured to be Large. This is done by driving 82496 CLDRV[BGT#] (pin N04) high during reset. 82496 and 82491 Memory Bus buffer sizes must be controlled by the Memory Bus Controller.
5. Series termination resistors were added to the nets PA17, PA18, PA19, and PA20 to control overshoot. A value of 24 Ω is currently recommended, but that value may change when overshoot is measured on an actual board.

7.6 Explanation of Information Provided

The following sections outline the design files associated with the 256Kbyte CPU-Cache Chip Set design example that are available from Intel. These files are provided to simplify the task of porting the design example

into a specific design. By using these files, designers may eliminate or minimize the amount of duplicate effort when using the design example as the basis for their design. The following items are available:

- Schematics
- I/O Model Files
- Board Files
- Bill of Materials
- Photoplot Log
- Netlist Report
- Placed Component Report
- Artwork for Each Board Layer
- Trace Segment Line Lengths

Hard copies of the schematics and trace segment line lengths are provided in the following sections. ASCII or soft copies of all the information are available from Intel by requesting order number 241663, *AP-481 Design Diskettes*.

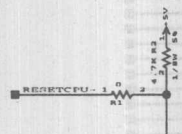
7.6.1 SCHEMATICS

Schematics for the 256 Kbyte CPU-Cache Chip Set design example were created using ViewLogics's Workview V4.1. The schematics are 14 pages long. Both the Workview and the postscript files are available from Intel as described above.

Pentium™ Processor/82496/82491 256 KB OPTIMIZED INTERFACE 1-20-92 REV 2.1

NOTES: (Unless Otherwise Specified)

1. Capacitor values are in microfarads.
2. Resistor values are in ohms.
3. An "-" following a signal name denotes negation.
4. VCC = +5V.
5. This document also exists on electronic media.



CD(0.63)

CP(0.7)

Pentium Processor PA33									
CD0	87	D3	T17	A3	204	PA3			
CD1	88	D1	E3	W18	A4	271	PA4		
CD2	89	D2	E4	U18	A5	228	PA5		
CD3	90	D3	F3	U17	A6	227	PA6		
CD4	91	D4	F4	T18	A7	205	PA7		
CD5	92	D5	G3	U16	A8	226	PA8		
CD6	93	D6	E4	T18	A9	204	PA9		
CD7	94	D7	G4	U15	A10	225	PA10		
CD8	95	D8	F4	T14	A11	203	PA11		
CD9	96	D9	C12	U14	A12	224	PA12		
CD10	97	D10	C13	T13	A13	202	PA13		
CD11	98	D11	E5	U13	A14	221	PA14		
CD12	99	D12	G4	T12	A15	201	PA15		
CD13	100	D13	G4	U12	A16	222	PA16		
CD14	101	D14	D12	T11	A17	200	CPA17	1/2W R5	PA17
CD15	102	D15	D6	U11	A18	221	CPA18	1/2W R6	PA18
CD16	103	D16	D6	T10	A19	199	CPA19	1/2W R6	PA19
CD17	104	D17	E9	U10	A20	220	CPA20	1/2W R6	PA20
CD18	105	D18	E6	U21	A21	219	CPA21	1/2W R6	PA21
CD19	106	D19	C15	U9	A22	218	PA22		
CD20	107	D20	D7	U20	A23	216	PA23		
CD21	108	D21	C16	U8	A24	218	PA24		
CD22	109	D22	C7	U19	A25	229	PA25		
CD23	110	D23	A10	T9	A26	198	PA26		
CD24	111	D24	B10	V21	A27	251	PA27		
CD25	112	D25	C8	V6	A28	237	PA28		
CD26	113	D26	C11	V20	A29	231	PA29		
CD27	114	D27	D9	W5	A30	257	PA30		
CD28	115	D28	D11	V19	A31	250	PA31		
CD29	116	D29	C9	T8	BT0	197	PBT0		
CD30	117	D30	D12	W21	BT1	273	PBT1		
CD31	118	D31	D10	T7	BT2	196	PBT2		
CD32	119	D32	D10	W20	BT3	272	PBT3		
CD33	120	D33	C17	T5					
CD34	121	D34	C19						
CD35	122	D35	D17						
CD36	123	D36	C18						
CD37	124	D37	D16						
CD38	125	D38	D19						
CD39	126	D39	D18						
CD40	127	D40	D14						
CD41	128	D41	B19						
CD42	129	D42	D20						
CD43	130	D43	A20						
CD44	131	D44	D21						
CD45	132	D45	A21						
CD46	133	D46	E18						
CD47	134	D47	B20						
CD48	135	D48	B21						
CD49	136	D49	F19						
CD50	137	D50	C20						
CD51	138	D51	F18						
CD52	139	D52	C21						
CD53	140	D53	G18	M3	FERR	112	FERR		
CD54	141	D54	E20	V2	BREQ	233	BREQ		
CD55	142	D55	G19	M4	HITH	145	HITH		
CD56	143	D56	H21	W2	HIT	144	HIT		
CD57	144	D57	F20	M3	PCHK	176	PCHK		
CD58	145	D58	J18	E3	AP	160	AP		
CD59	146	D59	H10	G3	HLDA	168	HLDA		
CD60	147	D60	L19	V3	LOCK	234	LOCK		
CD61	148	D61	B19	M3	APCHK	255	APCHK		
CD62	149	D62	J19	U3	PRDY	111	PRDY		
CD63	150	D63	H18	M4	ADSC	183	ADSC		
CP0	151	CP0	H4	F4	ADR	161	ADR		
CP1	152	CP1	C6	M3	WR	182	WR		
CP2	153	CP2	A9	V4	PCD	256	PCD		
CP3	154	CP3	D8	E3	PWT	184	PWT		
CP4	155	CP4	D18	V4	DC	255	DC		
CP5	156	CP5	A19	U4	BE0	214	CBE0		
CP6	157	CP6	B19	G4	BE1	189	CBE1		
CP7	158	CP7	E21	U6	BE2	216	CBE2		
CLK0	159	CLK0	H18	V1	BE3	233	CBE3		
HOLD	210	HOLD	V5	T6	BE4	195	CBE4		
BOFF	129	BOFF	K4	G4	BE5	185	CBE5		
A20H	215	A20H	U5	U7	BE6	217	CBE6		
INTR	154	INTR	H18	W1	BE7	253	CBE7		
NMI	155	NMI	H19	M4	SCYC	177	SCYC		
ICNNE	188	ICNNE	E20						
KEN	120	KEN	J3	M19	FRMC	147	FRMC		
HC	212	HC	U2						
ANOLD	131	ANOLD	L3						
EADE	144	EADE	M3						
BRDY0	137	BRDY0	L4	E21	TDICB1	189	TDICB1		
RESETCPU	133	RESETCPU	L18	T19	IBT	208	IBT		
BRDYC1	136	BRDYC1	L3	C2	IERR	144	IERR		
INV	129	INV	A1	T5	SMIACT	194	SMIACT		
NA	128	NA	K3	D2	PHBPO	15	PHBPO		
BUSCHK	120	BUSCHK	T3	C3	PHBPI	15	PHBPI		
TCK	193	TCK	T4	B2	BP1	23	BP1		
SWRE	30	SWRE	A3	B3	BP2	24	BP2		
WBWT	143	WBWT	M2	A2	MIO	2	MIO		
INITCPU	203	INITCPU	T20	J4	CACHE	121	CACHE		
SMI	162	SMI	F18	B1	IV	23	IV		
RS	172	RS	H18	J2	IO	119	IO		
TMS	161	TMS	F19						
TRST	186	TRST	H18						
PEN	146	PEN	H18						
TDICPU	210	TDICPU	T21	Q19					

PA(3:31)

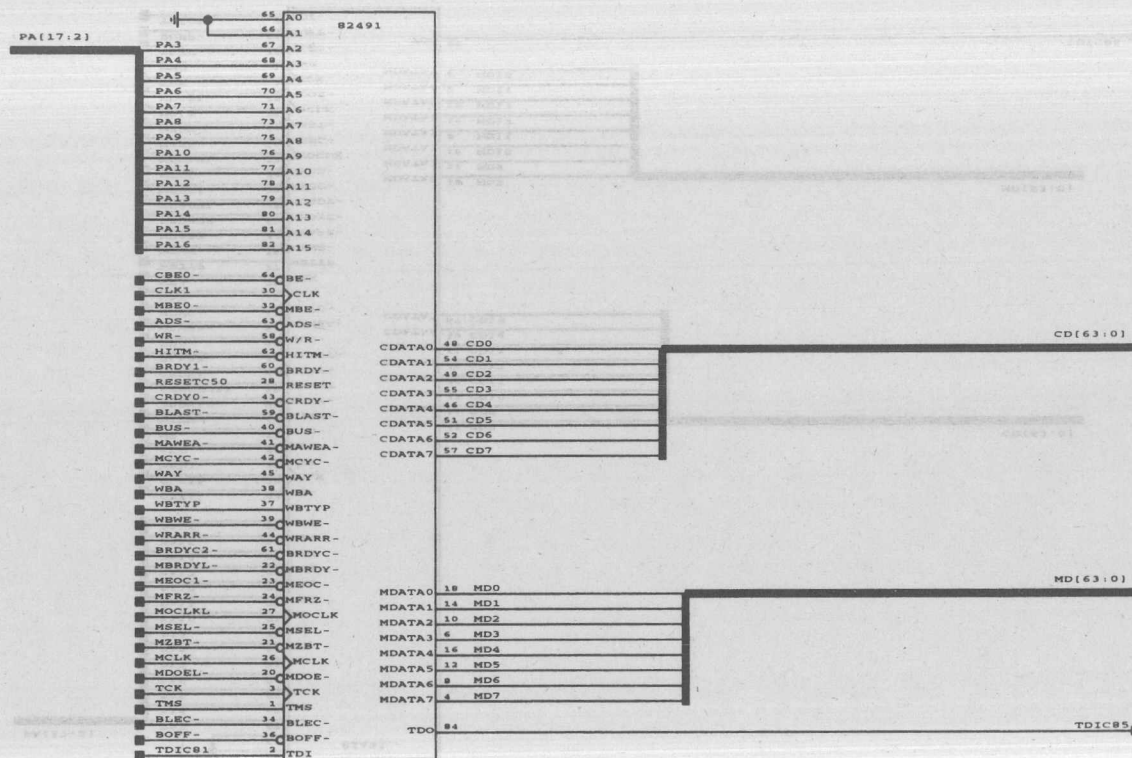
3

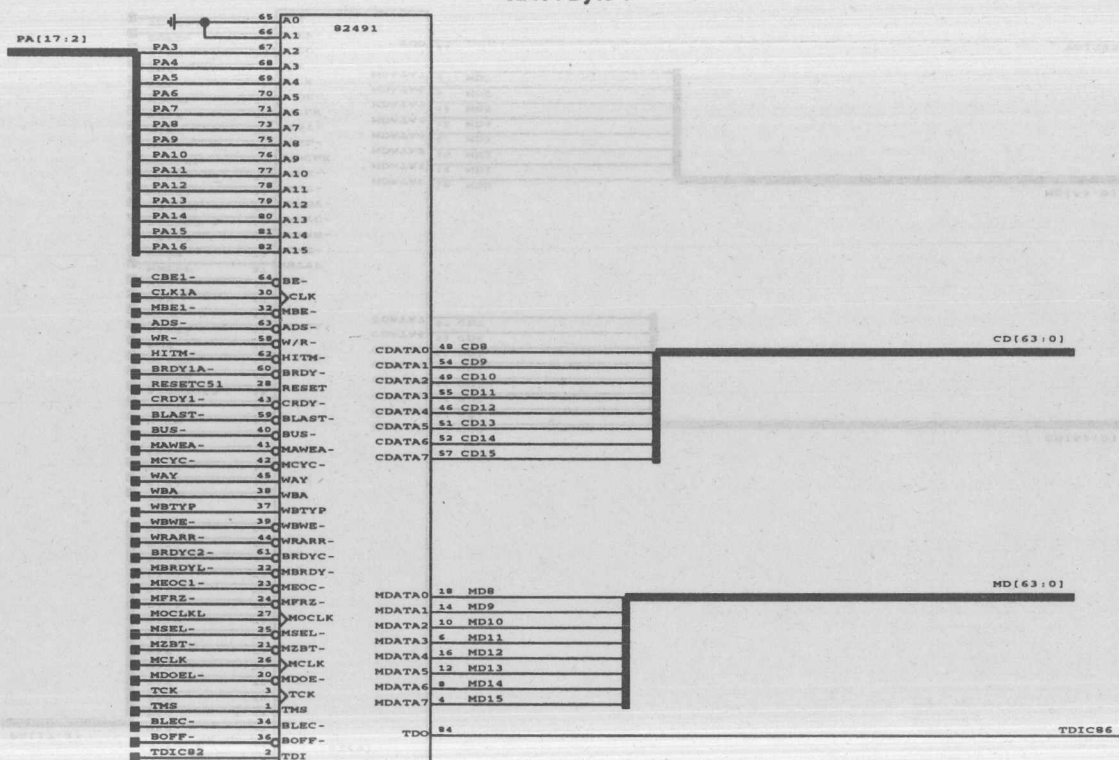
#2496									
PA3 102	CFA0	F16 R17	MCPAD	221	MA3				
PA4 83	CFA1	C16 P16	MCPA1	181	MA4				
PA29 82	CFA2	E7 R6	MCPA2	228	MA29				
PA30 41	CFA3	C3 R7	MCPA3	211	MA30				
PA31 36	CFA4	B17 Q16	MCPA4	201	MA31				
103	CFA5	F18 Q16	MCPA5	200					
PA5 81	CFA6	E16 Q14	MCPA6	199	MA5				
PBT0 16	BT0	A16 U14	MBT0	277	MBT0				
PBT1 14	BT1	A14 U14	MBT1	276	MBT1				
PBT2 12	BT2	A12 U12	MBT2	273	MBT2				
PBT3 10	BT3	A10 U10	MBT3	271	MBT3				
AP 8	AP	A8							
CSCYC 77	CSCYC	E1							
SMLN- 54	SMLN-	D7 R16	MBT0	220	MA6				
PA6 90	SET0	E14 Q13	MBT1	198	MA7				
PA7 73	SET1	D14 Q12	MBT2	197	MA8				
PA8 72	SET2	D15 R15	MBT3	218	MA9				
PA9 51	SET3	C13 R17	MBT4	240	MA10				
PA10 70	SET4	D13 R14	MBT5	218	MA11				
PA11 69	SET5	D12 R18	MBT6	241	MA12				
PA12 82	SET6	E13 T18	MBT7	260	MA13				
PA13 86	SET7	E10 Q11	MBT8	196	MA14				
PA14 87	SET8	E11 R13	MBT9	217	MA15				
PA15 48	SET9	C10 R12	MBT10	216	MA16				
PA16 29	SET10	B10 R11	MTAG0	215	MA17				
PA17 66	TAG0	D9 Q10	MTAG1	195	MA18				
PA18 24	TAG1	B5 R10	MTAG2	214	MA19				
PA19 85	TAG2	E9 R15	MTAG3	238	MA20				
PA20 23	TAG3	B4 R9	MTAG4	213	MA21				
PA21 43	TAG4	C5 R16	MTAG5	239	MA22				
PA22 42	TAG5	C4 T15	MTAG6	257	MA23				
PA23 55	TAG6	D8 T16	MTAG7	258	MA24				
PA24 22	TAG7	B3 Q9	MTAG8	194	MA25				
PA25 84	TAG8	E8 T17	MTAG9	250	MA26				
PA26 21	TAG9	B2 R8	MTAG10	212	MA27				
PA27 40	TAG10	C2 Q8	MTAG11	193	MA28				
PA28 63	TAG11	D6 R18	AHOLD	27	AHOLD				
ADSC- 54	ADR-	C16 R16	EADS-	142	EADS-				
WR- 56	W/R-	C18 R17	KEN-	93	KEN-				
DC- 131	D/C-	J15 E16	BRDYC1-	92	BRDYC1-				
MIO- 113	M/IO-	G17 R18	NA-	144	NA-				
PCD 132	PCD	J16 D17	BLE-	74	NC				
PWT 75	PWT	D18 Q15	BRDYC2-	111	BRDYC2-				
LOCK- 55	LOCK-	C17 Q17	BUS-	202	BUS-				
SCYC 112	SCYC	G16 Q18	MCYC-	203	MCYC-				
WBWT- 151	WB/WT-	L15 R18	MAWEA-	222	MAWEA-				
CACHE- 121	CACHE-	H15 R16	WAY	162	WAY				
CLKO 88	CLK	E12 R15	WBA	171	WBA				
TDI 179	TDI	P4 P16	WBTP	182	WBTP				
TMS 188	TMS	Q3 R16	WBWE-	172	WBWE-				
TCK 189	TCK	Q4 R15	WRARR-	141	WRARR-				
		D16	BLAST-	71	BLAST-				
HITM- 94	HITH-	E18 F4	CADS-	99	CADS-				
BRDY0- 187	BRDY-	Q2 G4	SNPADS-	109	SNPADS-				
RESETC50 210	RESET	R6 G5	CDTS-	110	CDTS-				
CRDY0- 168	CRDY-	H3 F5	CW/R-	100	CWR-				
C5FLUSH- 180	FLUSH-	F5 E4	CD/C-	80	CDC-				
BGT- 169	BGT-	N4 E5	CM/IO-	81	CMIO-				
CNA- 160	CNA-	H5 D2	RDYSRC	59	RDYSRC				
SWEND- 206	SWEND-	R2 D3	MCACHE-	60	MCACHE-				
KWEND- 170	KWEND-	H5 D4	KLOCK-	61	KLOCK-				
PCYC 141	PCYC	K15 H5	CAHOLD	120	CAHOLD				
SYNC- 209	SYNC-	R5 E3	PALLC-	79	PALLC-				
MKEN- 225	MKEN-	B2 K4	CWAY	139	CWAY				
MRO- 137	MRO-	K2 E6	NENE-	82	NENE-				
MWBWT- 149	MWB/WT-	L4							
DRCTM- 167	DRCTN-	N2 H16	BOFF-	122	BOFF-				
MAOE- 147	MAOE-	T5 L16	INV	152	INV				
MAOE- 192	MAOE-	Q7 E2	FSIOUT-	78	FSIOUT-				
MALE 207	MALE	R3							
MALE 190	MALE	Q5 J5	MHITM-	130	MHITM-				
SNPINV 191	SNPINV	Q6 H4	MTHIT-	119	MTHIT-				
SNPNCA 208	SNPNCA	R4 J4	SNPCYC-	120	SNPCYC-				
SNPSTB- 227	SNPSTB-	B4 G2	SNPBY-	107	SNPBY-				
SNPCLK 246	SNPCLK	T4 D5	TDICPU	62	TDICPU				
CCACHE- 136	CCACHE-	H1							
TRST- 244	TRST-	T2 K1	CPWT	136	CPWT				
MAP 269	MAP	U8 M1	CPCD	156	CPCD				
		N1	MPIC-	166	MPIC-				
		P19	BLEC-	185	BLEC-				
		Q1	IPERR-	186	IPERR-				
		S1	APERR-	224	APERR-				
		T3	EWBE-	245	EWBE-				
		U1	MAPERR-	262	MAPERR-				

U2

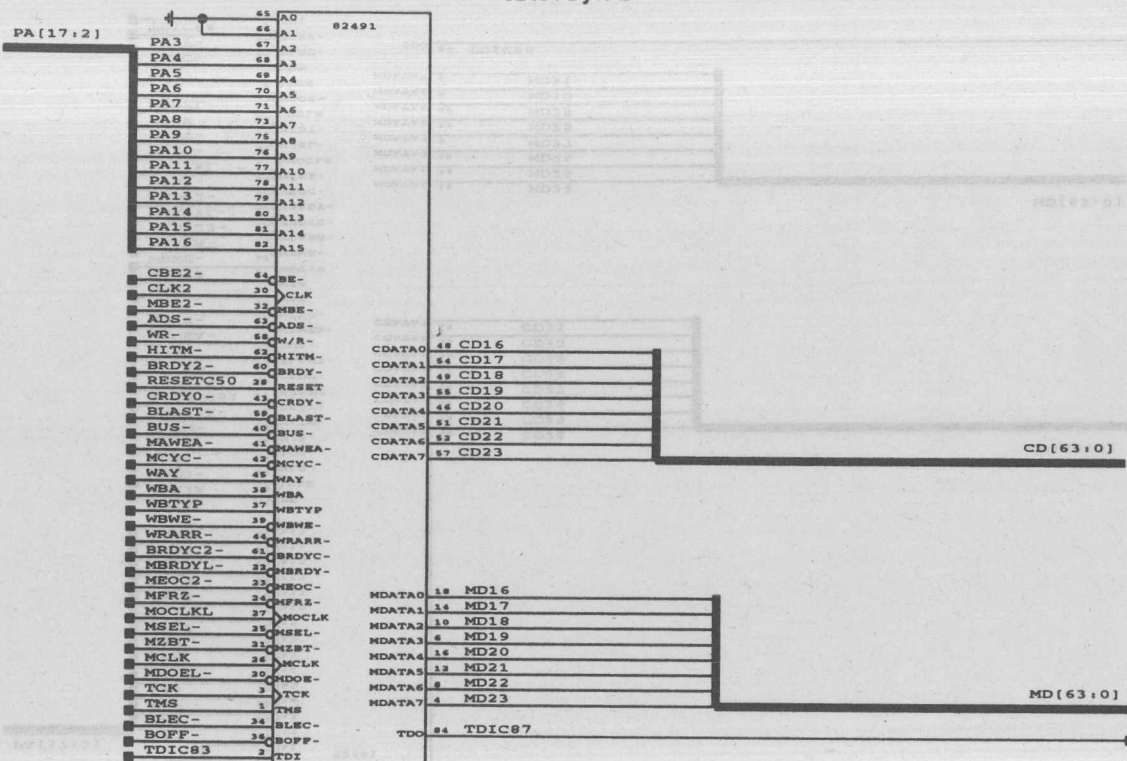
241576-68

241576-69





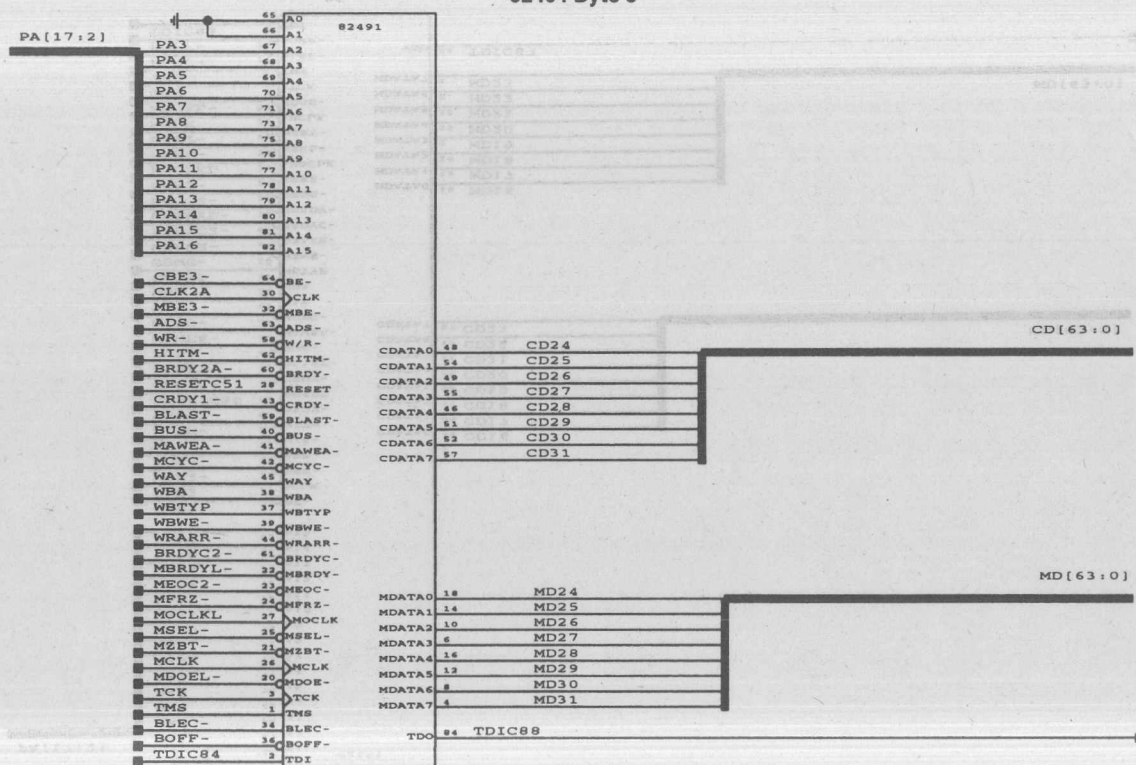
82491 Byte 2



U5

241576-71

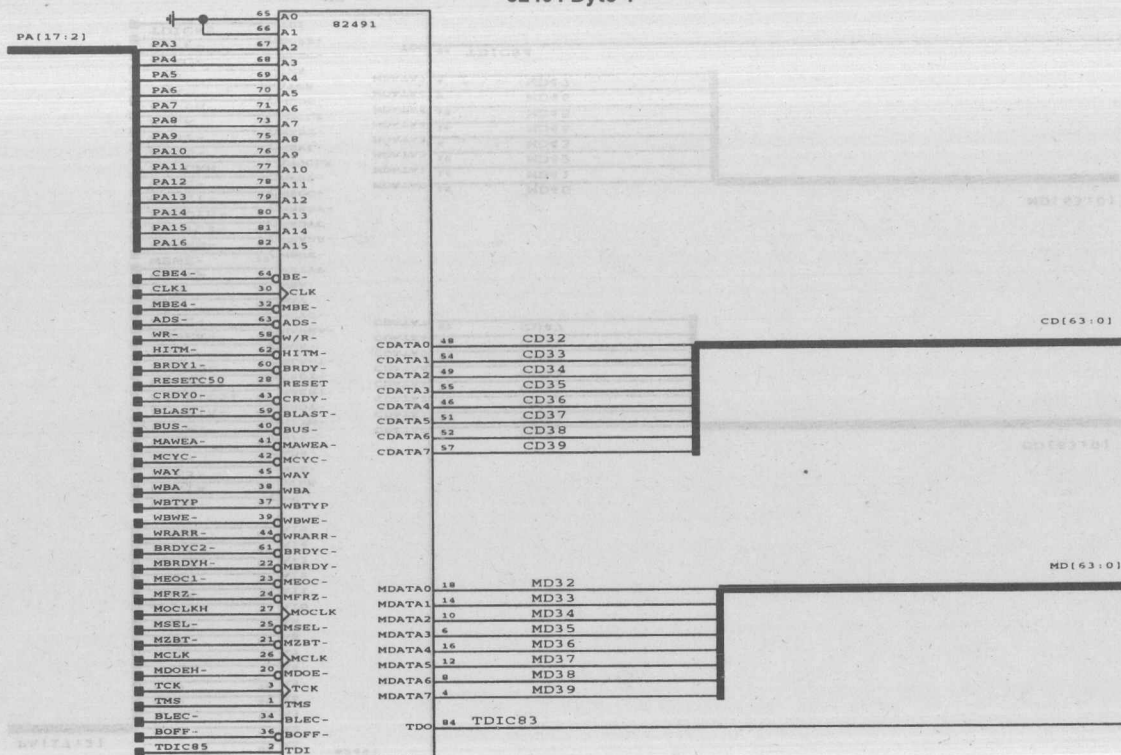
82491 Byte 3



U6

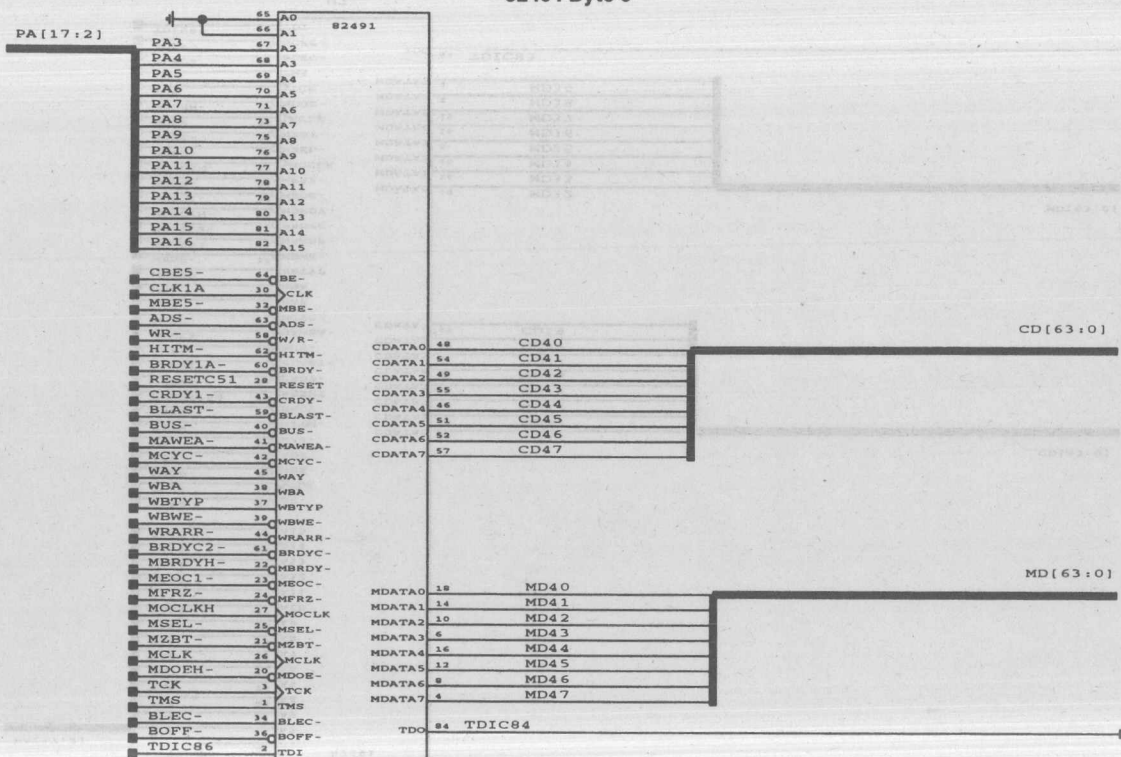
241576-72

82491 Byte 4



241576-73

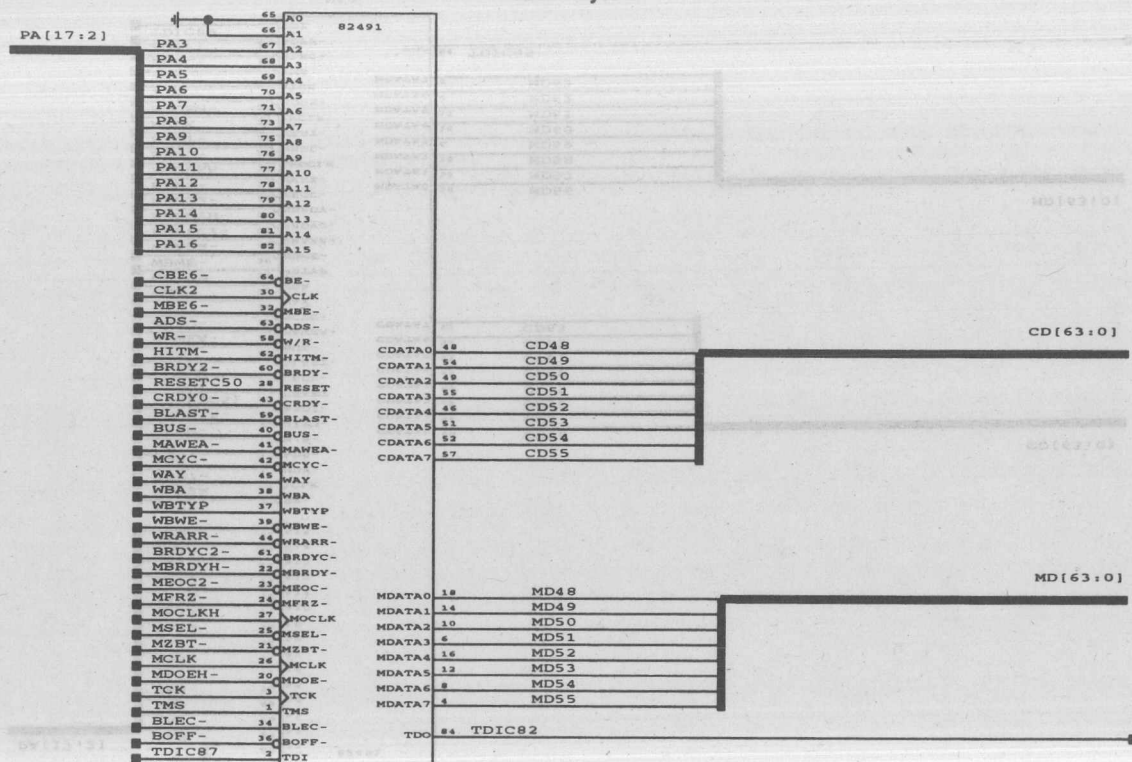
82491 Byte 5



U8

241576-74

82491 Byte 6

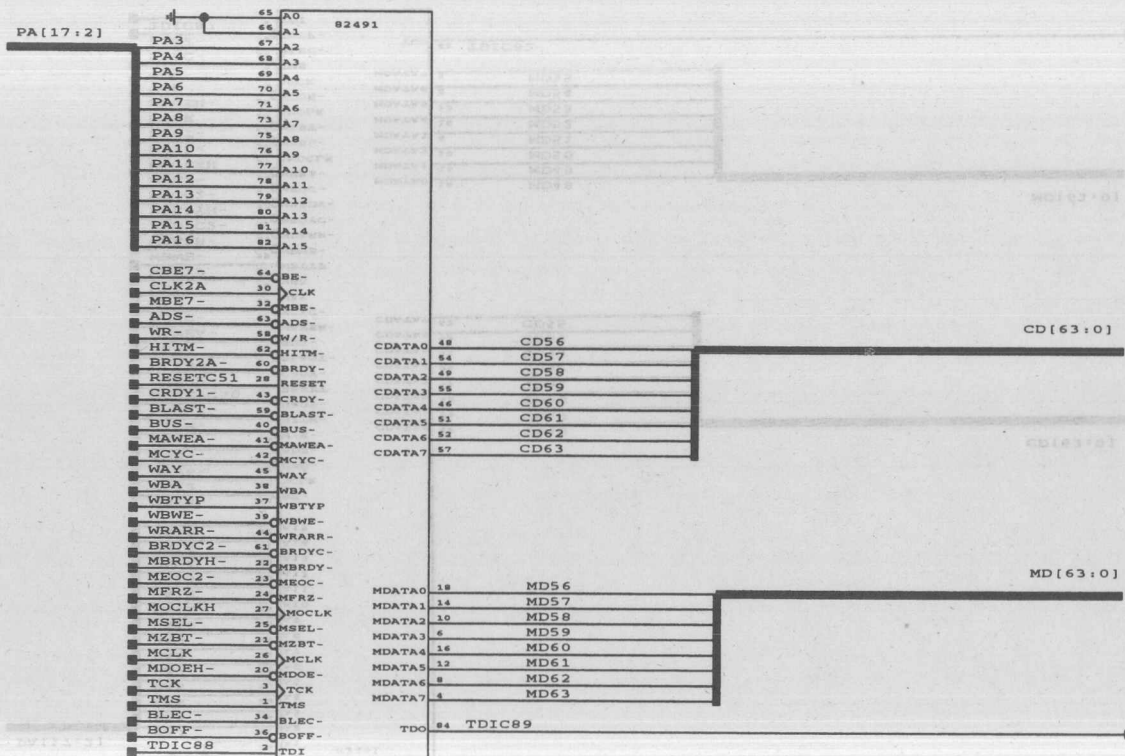


U9

241576-75

82491 Byte 7

344210-12

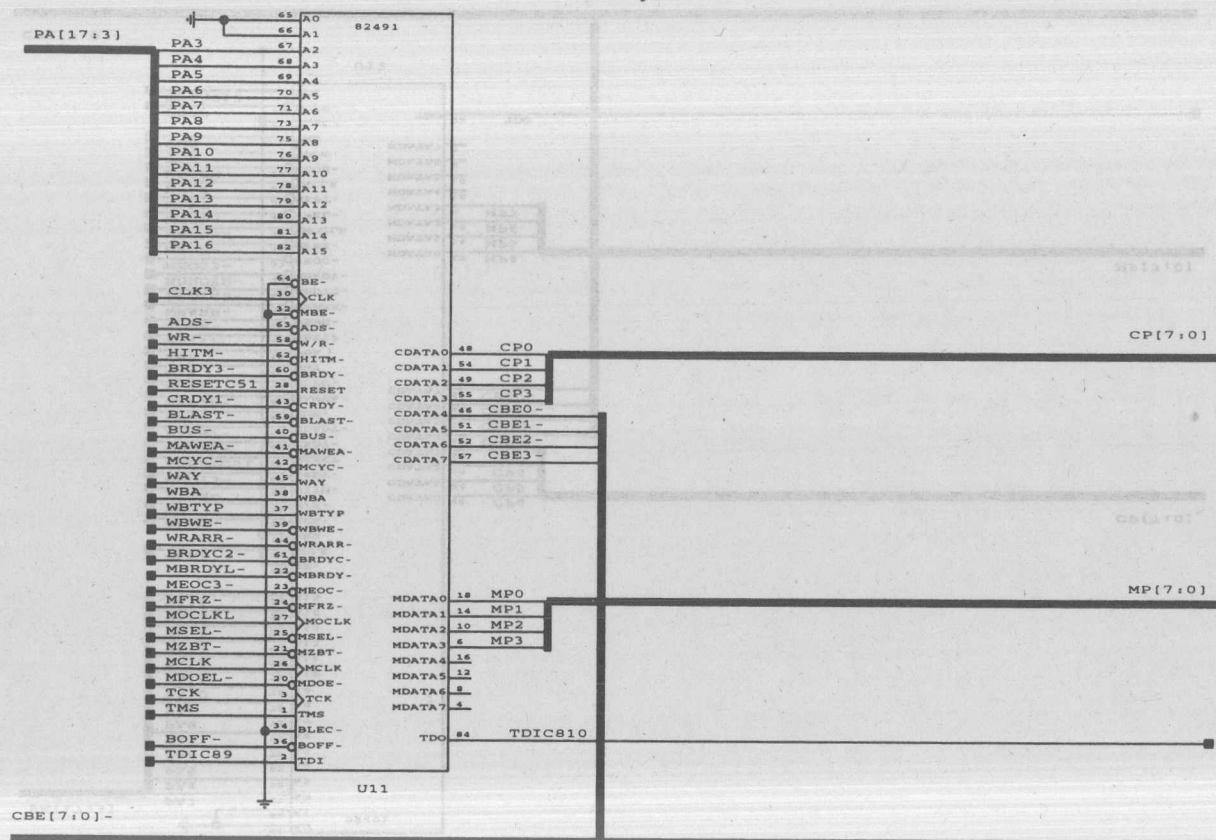


U10

344210-12

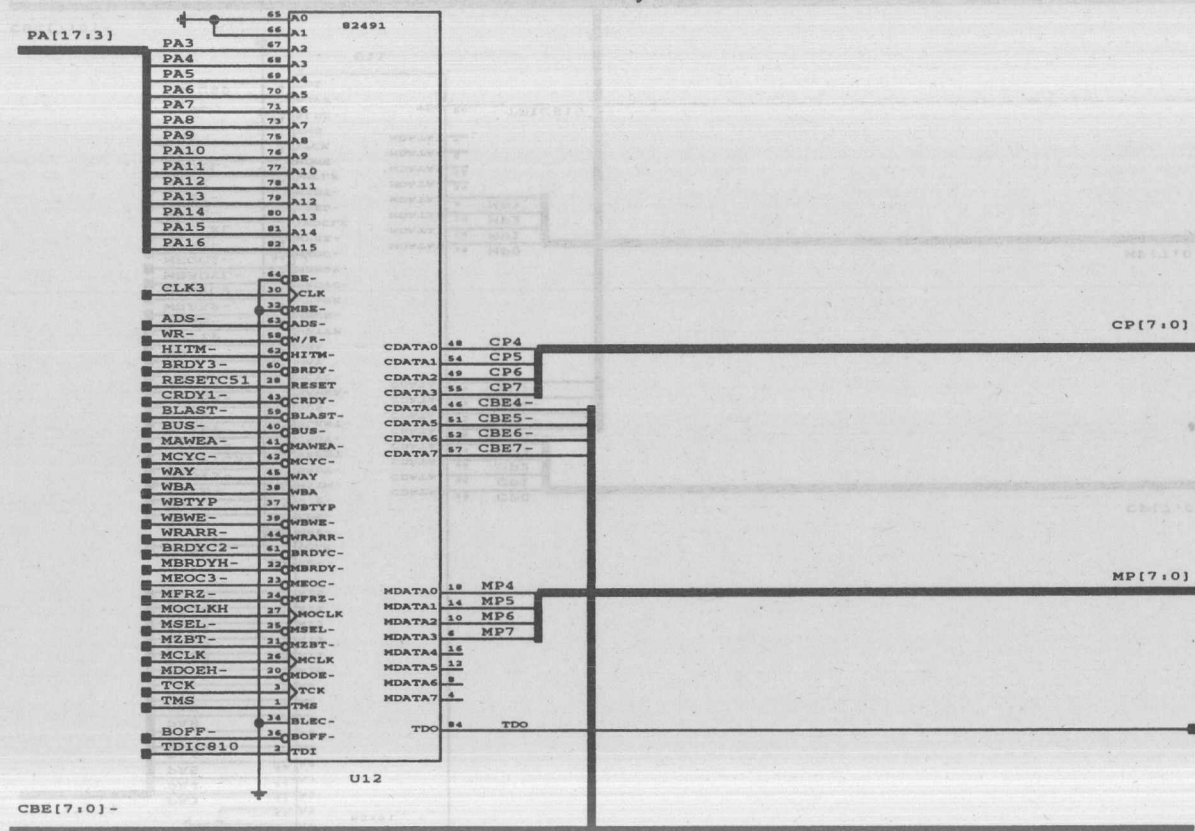
241576-76

82491 Parity 0-3



241576-77

82491 Parity 4-7



241576-78

7.6.2 I/O MODEL FILES

All electrical I/O simulations were performed using TLC V4.1.13 from Quad Design Technology, Inc. The simulations were performed at the fast and slow corners to verify all signal quality and flight time specifications are met. The files used for these simulations are available from Intel as described above. These files include the topology, model, and control files needed to run the simulations for all nets in the optimized interface.

7.6.3 BOARD FILES

The board files for the design example were created using Allegro V4.2 from Cadence Design Systems, Inc. The files are available from Intel as described above. These files may be used to import the design example into a specific system design. Note: some changes to the layout and nets may be necessary to complete importing these files into a specific system design.

7.6.4 BILL OF MATERIALS

The bill of materials file was created using Allegro V4.2 from Cadence Design Systems, Inc. The file is available from Intel as described above.

7.6.5 PHOTOPLOT LOG

The photoplot log file was created using Allegro V4.2 from Cadence Design Systems, Inc. The file is available from Intel as described above.

7.6.6 NETLIST REPORT

The netlist report was created using Allegro V4.2 from Cadence Design Systems, Inc. The file is available from Intel as described above.

7.6.7 PLACED COMPONENT REPORT

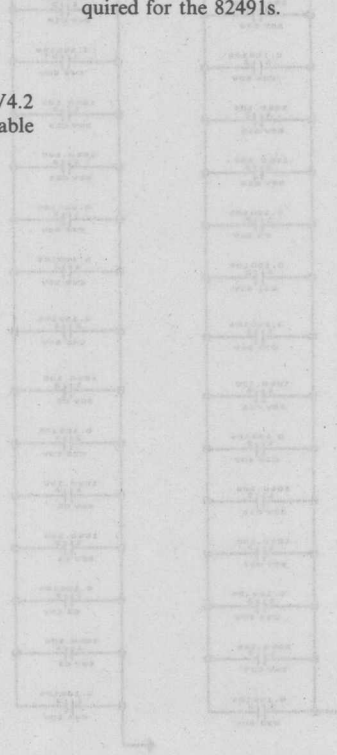
The placed component report was created using Allegro V4.2 from Cadence Design Systems, Inc. The file is available from Intel as described above.

7.6.8 ARTWORK FOR EACH BOARD LAYER

The artwork for the six board layers were created using Allegro V4.2 from Cadence Design Systems, Inc. The files are available from Intel in a Gerber format as described above.

7.6.9 TRACE SEGMENT LINE LENGTHS

Sections 7.6.9.1 to 7.6.9.10 list the segment line lengths for each net of the optimized interface. All lengths are provide in mils (1/1000 inch). The stubs listed in the following tables are associated with the pin escapes required for the 82491s.



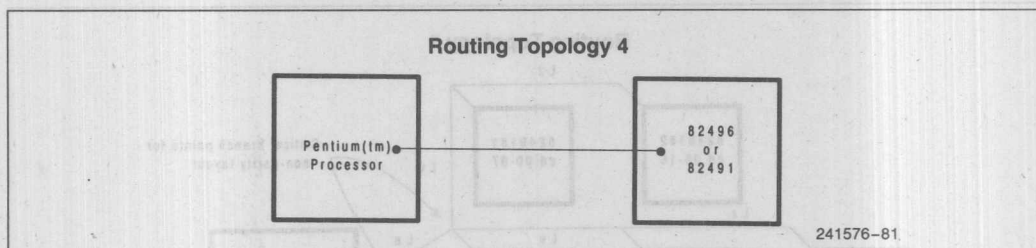
NET	PP-CS #5	PP-CS #3	PP-CC	CS #5-CS #1	CS #1-CS #2	CS #5-CS #6	CS #6-CS #9
PA3	1761.4	1768.6	5513	1184.6	936.5	1186.5	936.5
PA4	1259.6	1278.4	4673.2	1113.9	936.5	1113.6	936.5
PA5	1553.6	1573.9	5193.7	1164.6	936.5	1176.5	936.5
PA6	1543.1	1540	5123.2	1152.9	936.5	1156.5	936.5
PA7	1691.9	1692.4	5367.6	1152.9	936.5	1156.5	936.5
PA8	1590.7	1590.2	5243.7	1215.3	936.5	1216.5	936.5
PA9	1660.7	1663.1	5365.2	1210.5	936.5	1206.5	936.5
PA10	1543.6	1543.1	5233	1264.1	936.5	1266.5	936.5
PA11	1701.9	1695.5	5474.2	1264.1	936.5	1266.5	936.5
PA12	1586.5	1594.3	5324.1	1292.4	936.5	1296.5	936.5
PA13	1741.9	1745.5	5497.9	1295.3	936.5	1296.5	936.5
PA14	1633.6	1633.1	5403.8	1317.8	936.5	1316.5	936.5
PA15	1791.9	1792.6	5500.4	1314.8	936.5	1316.5	936.5
PA16	1623.6	1619	5359.1	1288.0	936.5	1286.5	936.5

PP = Pentium processor
 CC = 82496 cache controller
 CS = 82491 cache SRAM

NET	CS #3-CS #4	CS #4-CS #10	CS #3-CS #7	CS #7-CS #8	Stubs
PA3 (cont)	1181.4	936.5	1184.6	936.5	135.3-135.3
PA4 (cont)	1111.4	936.5	1113.9	936.5	75.0-75.0
PA5 (cont)	1166.5	936.5	1170.5	936.5	135.3-135.3
PA6 (cont)	1156.5	936.5	1158.8	936.5	75.0-75.0
PA7 (cont)	1156.5	936.5	1158.8	936.5	135.3-135.3
PA8 (cont)	1216.5	936.5	1212.4	936.5	135.3-135.3
PA9 (cont)	1206.5	936.5	1207.6	936.5	135.3-135.3
PA10 (cont)	1266.5	936.5	1261.2	936.5	75.0-75.0
PA11 (cont)	1266.5	936.5	1261.2	936.5	135.3-135.3
PA12 (cont)	1296.5	936.5	1292.4	936.5	75.0-75.0
PA13 (cont)	1296.5	936.5	1295.3	936.5	135.3-135.3
PA14 (cont)	1316.5	936.5	1317.8	936.5	75.0-75.0
PA15 (cont)	1316.5	936.5	1314.8	936.5	135.3-135.3
PA16 (cont)	1286.5	936.5	1288.0	936.5	75.0-75.0

PP = Pentium processor
 CC = 82496 cache controller
 CS = 82491 cache SRAM

7.6.9.2 High Addresses (Topology 4, Point-to-Point)



NET	PP-CC
PA17*	689.3 + 2841.7
PA18*	647.6 + 3683.1
PA19*	731.0 + 2763.3
PA20*	601.7 + 718.6
PA21	3376.6
PA22	4347.5
PA23	3111.5
PA24	4661.2
PA25	3029.7

NET	PP-CC
PA26	4495.1
PA27	3885.5
PA28	4689.3
PA29	3136.1
PA30	5177.1
PA31	2518.5
PA32	3604.4
PA33	2686.8
PA34	3952
PA35	3189.9

NET	PP-CC
ADSC#	3791.7
AP	4531.6
CACHE#	3719.8
DC#	3613.1
LOCK#	4710.4
MIO#	5062
PCD	3461.3
PWT	4295.6
SCYC	3848.2
WBWT#	3493.3

***NOTE:**

24Ω resistor included on PA[17-20].

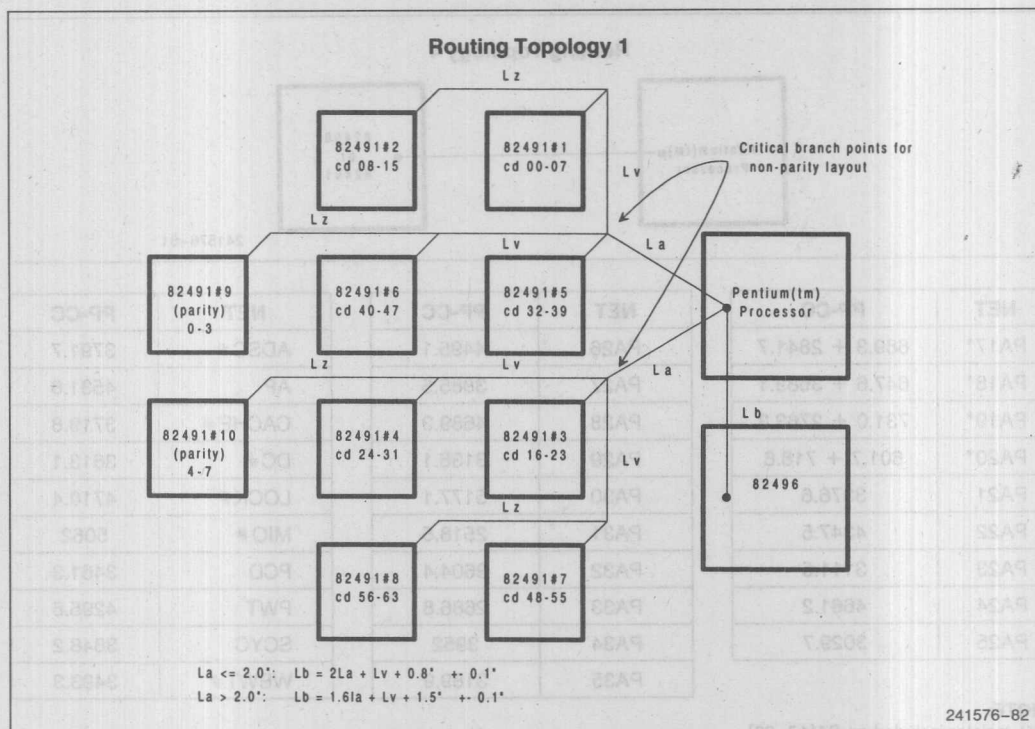
Lengths are Pentium processor-resistor + resistor-82486, respectively.

PP = Pentium processor

CC = 82496 cache controller

CS = 82491 cache SRAM

7.6.9.3 Pentium™ Processor Control (Topology 1)

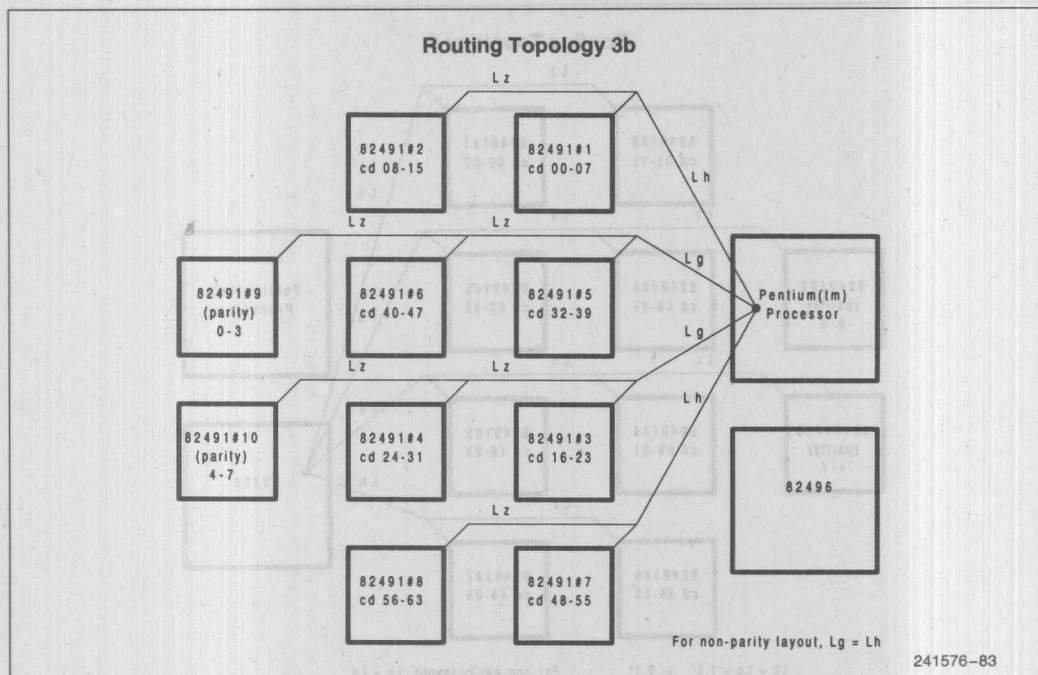


NET	PP-CS#5	PP-CS#3	PP-CC	CS#5-CS#1	CS#1-CS#2	CS#5-CS#6	CS#6-CS#9
HITM#	4205.2	4199.7	9147.1	1141.3	936.5	1146.5	936.5
WR#	4091.1	4088.2	9149.4	1193.0	936.5	1192.1	936.5

NET	CS#3-CS#4	CS#4-CS#10	CS#3-CS#7	CS#7-CS#8	Stubs
HITM# (cont)	1146.2	944.8	1146.6	944.8	95.3-95.3
WR# (cont)	1196.7	953.1	1193.0	953.1	95.3-95.3

PP= Pentium processor
 CC= 82496 cache controller
 CS= 82491 cache SRAM

7.6.9.4 Pentium™ Processor Control (Topology 3b No 82496)



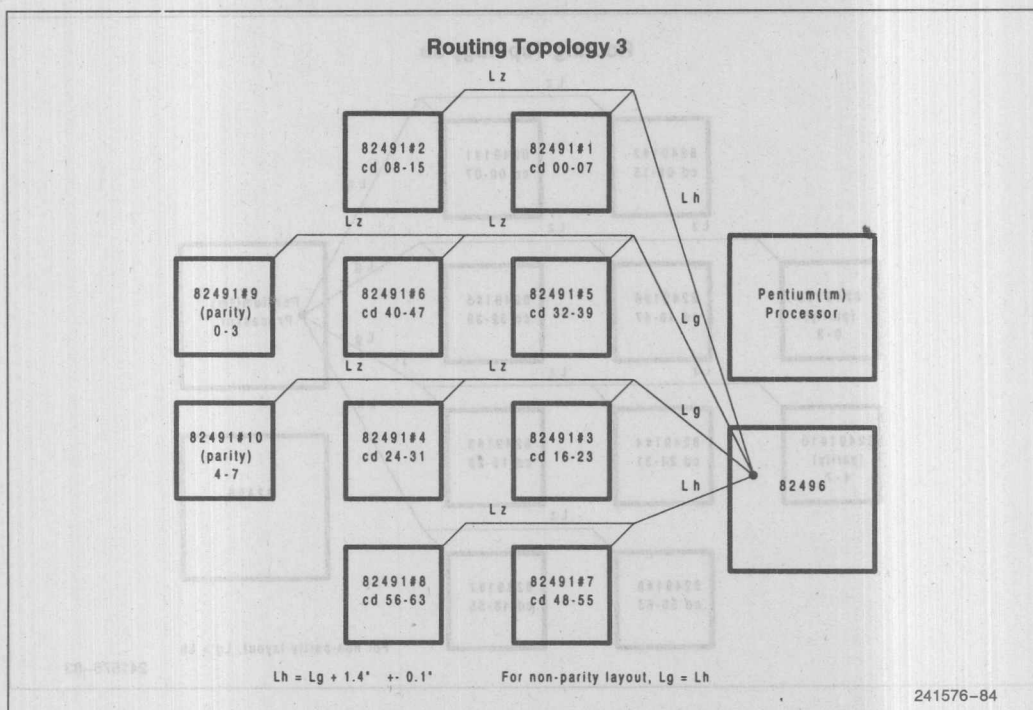
3

NET	PP-CS#1	PP-CS#5	PP-CS#3	PP-CS#7	CS#1-CS#2	CS#5-CS#6	CS#6-CS#9
ADS#	5113.0	3728.2	3738.5	5102.0	936.5	964.8	983.8

NET	CS#3-CS#4	CS#4-CS#10	CS#7-CS#8	Stubs
ADS# (cont)	936.5	936.5	936.5	75.0-75.0

PP= Pentium processor
CC= 82496 cache controller
CS= 82491 cache SRAM

7.6.9.5 82496 Control (Topology 3)

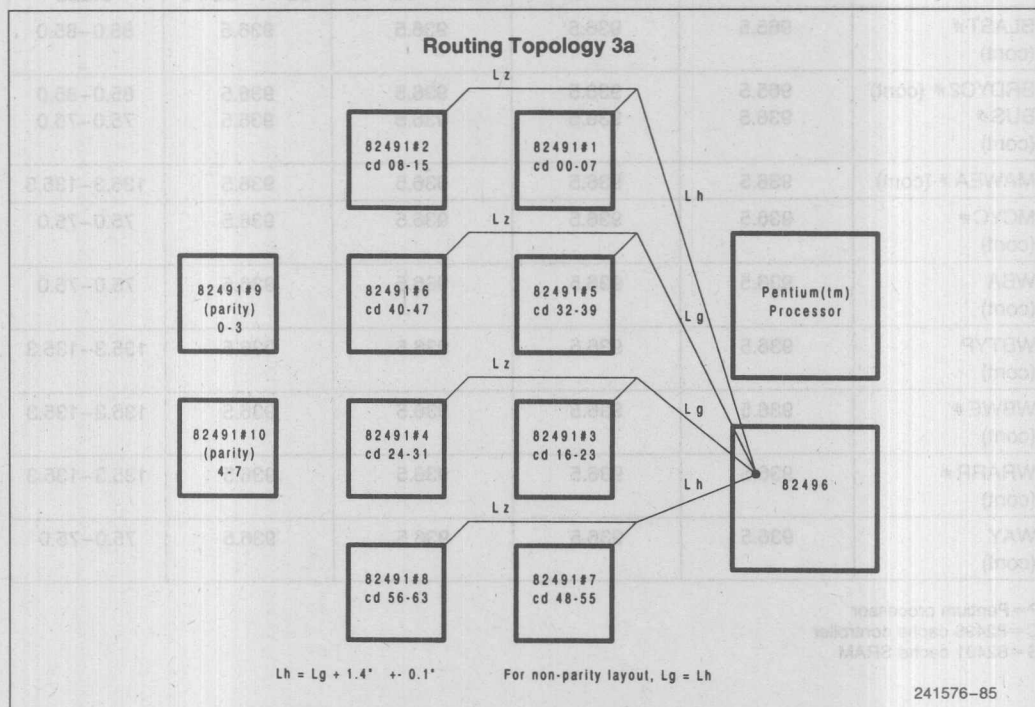


NET	CC-CS#1	CC-CS#5	CC-CS#3	CC-CS#7	CS#1-CS#2	CS#5-CS#6
BLAST#	4222.1	2820.0	2819.8	4219.2	986.7	1034.5
BRDYC2#	4186.4	2802.4	2775.0	4171.6	969.7	1037.4
BUS#	4607.6	3215.7	3210.0	4611.3	936.5	936.5
MAWEA#	4476.4	3058.9	3088.8	4481.3	936.5	936.5
MCYC#	4913.9	3507.0	3523.8	4921.5	936.5	961.4
WBA	5114.2	3747.8	3719.5	5119.3	936.5	936.5
WB Typ	4365.4	2986.2	2973.5	4376.0	936.5	936.5
WBWE#	4502.4	3109.1	3113.0	4511.7	936.5	936.5
WRARR#	4198.9	2735.0	2802.1	4199.8	936.5	969.7
WAY	4818.9	3348.4	3417.4	4816.3	936.5	936.5

NET	CS#6-CS#9	CS#3-CS#4	CS#4-CS#10	CS#7-CS#8	Stubs
BLAST# (cont)	965.5	936.5	936.5	936.5	85.0-85.0
BRDYC2# (cont)	965.5	936.5	936.5	936.5	85.0-85.0
BUS# (cont)	936.5	936.5	936.5	936.5	75.0-75.0
MAWEA# (cont)	936.5	936.5	936.5	936.5	135.3-135.3
MCYC# (cont)	936.5	936.5	936.5	936.5	75.0-75.0
WBA (cont)	936.5	936.5	936.5	936.5	75.0-75.0
WBYP (cont)	936.5	936.5	936.5	936.5	135.3-135.3
WBWE# (cont)	936.5	936.5	936.5	936.5	135.3-135.3
WRARR# (cont)	936.5	936.5	936.5	936.5	135.3-135.3
WAY (cont)	936.5	936.5	936.5	936.5	75.0-75.0

PP = Pentium processor
CC = 82496 cache controller
CS = 82491 cache SRAM

7.6.9.6 82496 Control (Topology 3a Not Connected to Parity 82491's)

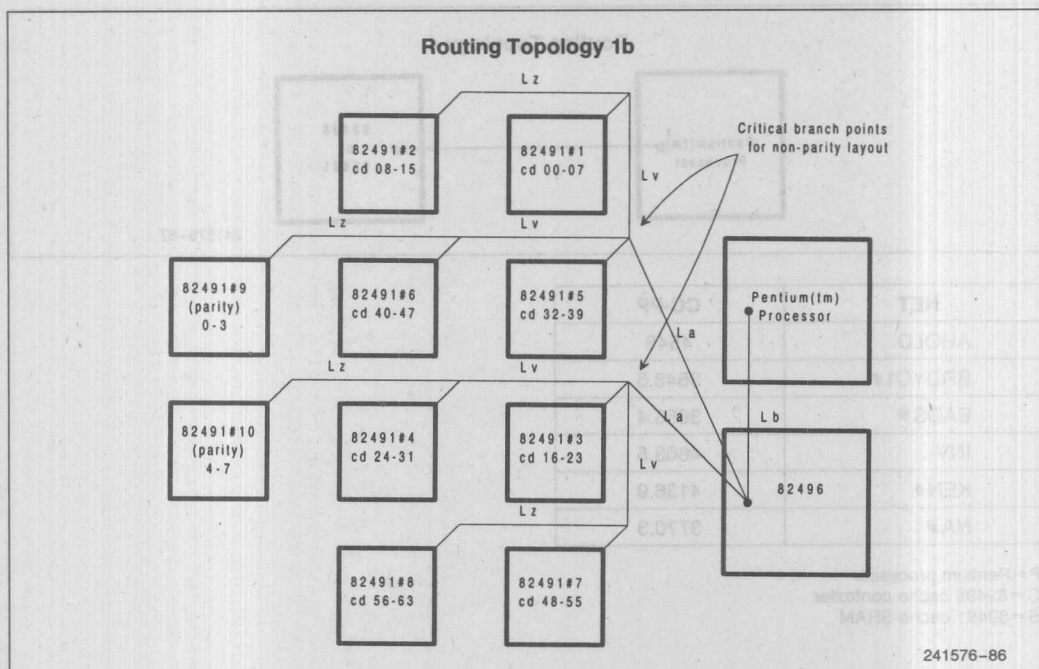


NET	CC-CS#1	CC-CS#5	CC-CS#3	CC-CS#7	CS#1-CS#2	CS#5-CS#6
BLEC#	4222.7	4219.0	4205.8	4206.4	936.5	936.5

NET	CS#6-CS#9	CS#3-CS#4	CS#4-CS#10	CS#7-CS#8	Stubs
BLEC# (cont)	n/a	936.5	n/a	936.5	75.0-75.0

PP=Pentium processor
 CC=82496 cache controller
 CS=82491 cache SRAM

7.6.9.7 82496 Control (Topology 1b Pentium™ Processor and 82496 Switch Positions)



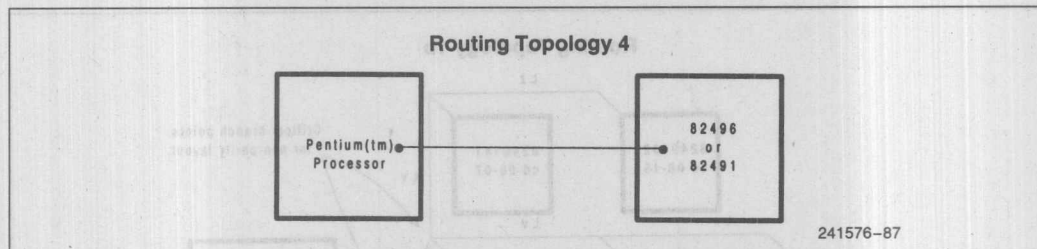
3

NET	CC-CS#5	CC-CS#3	CC-PP	CS#5-CS#1	CS#1-CS#2	CS#5-CS#6
BOFF#	3612.7	3596.1	7605.8	936.5	936.5	936.5

NET	CS#6-CS#9	CS#3-CS#4	CS#4-CS#10	CS#3-CS#7	CS#7-CS#8	Stubs
BOFF# (cont)	936.5	936.5	936.5	944.8	936.5	75.0-75.0

PP=Pentium processor
CC=82496 cache controller
CS=82491 cache SRAM

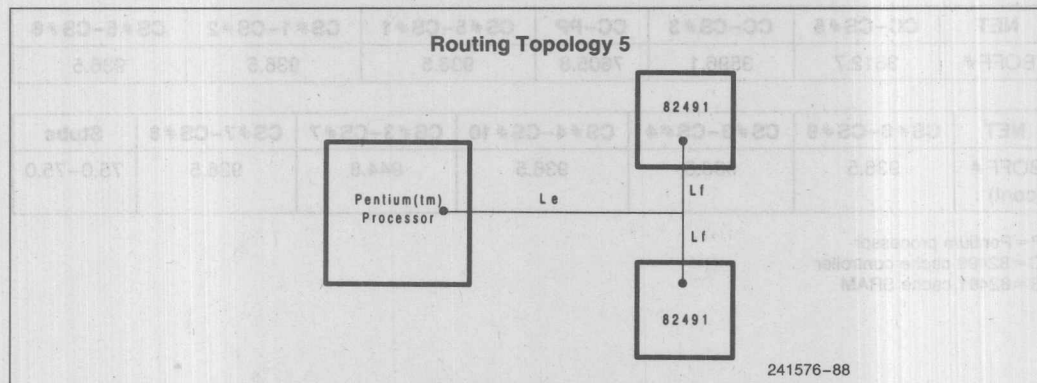
7.6.9.8 82496 Control (Topology 4, Point-to-Point)



NET	CC-PP
AHOLD	4549
BRDYC1 #	3648.5
EADS #	3656.4
INV	4603.5
KEN #	4136.9
NA #	3770.3

PP= Pentium processor
 CC= 82496 cache controller
 CS= 82491 cache SRAM

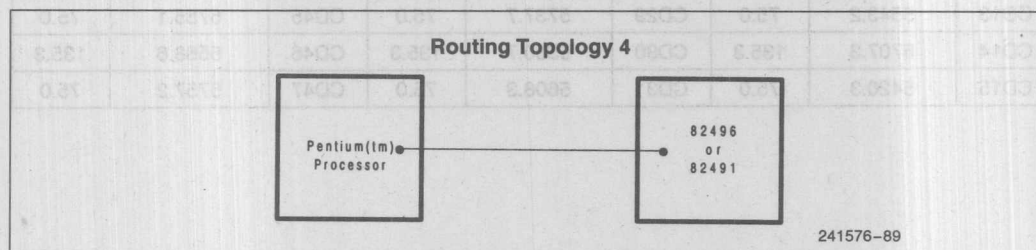
7.6.9.9 Byte Enables (Topology 5)



NET	PP-Tee	Tee-CS #1	Tee-CS #9	Stubs
CBE0 #	3035.4	1634.4	1633.9	112.4 135.3
NET	PP-Tee	Tee-CS #2	Tee-CS #9	Stubs
CBE1 #	4098.7	1294.8	1293.1	75.0-121.0
NET	PP-Tee	Tee-CS #3	Tee-CS #9	Stubs
CBE2 #	3412.9	1732.0	1682.3	75.0-95.3
NET	PP-Tee	Tee-CS #4	Tee-CS #9	Stubs
CBE3 #	3547.8	1194.8	1192.1	75.0-75.0
NET	PP-Tee	Tee-CS #5	Tee-CS #10	Stubs
CBE4 #	3600.9	2243.5	2242.0	75.0-135.3
NET	PP-Tee	Tee-CS #6	Tee-CS #10	Stubs
CBE5 #	4811.9	1339.7	1338.9	75.0-75.0
NET	PP-Tee	Tee-CS #7	Tee-CS #10	Stubs
CBE6 #	4662.0	1663.0	1662.6	75.0-135.3
NET	PP-Tee	Tee-CS #8	Tee-CS #10	Stubs
CBE7 #	4230.0	1167.7	1165.5	75.0-75.0

PP= Pentium processor
CC= 82496 cache controller
CS= 82491 cache SRAM

7.6.9.10 CData and Parity (Point-to-Point)



Net	PP-CS#9	Stub
CP0	5722.4	135.3
CP1	5842.1	135.3
CP2	5854.7	75.0
CP3	5712.4	75.0

NET	PP-CS#10	Stub
CP4	5831.1	135.3
CP5	5849.8	135.3
CP6	5563.8	75.0
CP7	5558.6	75.0

NET	PP-CS#1	Stub	NET	PP-CS#3	Stub	NET	PP-CS#5	Stub
CD0	5523.3	135.3	CD16	5541.4	135.3	CD32	5507.8	135.3
CD1	5376.2	135.3	CD17	5781.0	135.3	CD33	5419.5	135.3
CD2	5476.9	75.0	CD18	5550.7	75.0	CD34	5433.2	75.0
CD3	5363.3	75.0	CD19	5558.2	75.0	CD35	5756.3	75.0
CD4	5547.9	135.3	CD20	5449.7	135.3	CD36	5654.1	135.3
CD5	5393.5	75.0	CD21	5545.2	75.0	CD37	5551.6	75.0
CD6	5357.9	135.3	CD22	5410.4	135.3	CD38	5357.3	135.3
CD7	5582.3	75.0	CD23	5533.2	75.0	CD39	5451.6	75.0

NET	PP-CS#2	Stub	NET	PP-CS#4	Stub	NET	PP-CS#6	Stub
CD8	5448.4	135.3	CD24	5804.5	135.3	CD40	5430.0	135.3
CD9	5516.1	135.3	CD25	5594.4	135.3	CD41	5757.8	135.3
CD10	5629.1	75.0	CD26	5754.8	75.0	CD42	5378.1	75.0
CD11	5468.9	75.0	CD27	5705.4	75.0	CD43	5703.8	75.0
CD12	5451.5	135.3	CD28	5774.5	135.3	CD44	5462.3	135.3
CD13	5543.2	75.0	CD29	5737.7	75.0	CD45	5755.1	75.0
CD14	5707.3	135.3	CD30	5380.7	135.3	CD46	5568.6	135.3
CD15	5420.3	75.0	CD31	5608.3	75.0	CD47	5757.2	75.0

NET	PP-CS#7	Stub
CD48	5488.3	135.3
CD49	5551.8	135.3
CD50	5697.9	75.0
CD51	5581.1	75.0
CD52	5499.5	135.3
CD53	5704.4	75.0
CD54	5493.6	135.3
CD55	5627.9	75.0

PP = Pentium processor
 CC = 82496 cache controller
 CS = 82491 cache SRAM

7.6.10 Pentium™ PROCESSOR TO 82496 SEGMENT LENGTH AND ROUTING CHANGES

The example layout described in this application note was completed using early revisions to the I/O buffer models. This process was necessary to ensure that a board was available for the arrival of first silicon. After the models were improved based on the model validation and silicon characterization, the board layout was

NET	PP-CS#8	Stub
CD56	5553.9	135.3
CD57	5663.3	135.3
CD58	5602.7	75.0
CD59	5718.3	75.0
CD60	5451.6	135.3
CD61	5533.9	75.0
CD62	5550.4	135.3
CD63	5766.2	75.0

resimulated. These simulations have resulted in the recommendation to change the line length between the Pentium processor and 82496 for several nets. These changes result in a better tuned routing that meets the specifications. In particular, these changes reduce the amount of ringback and the ringing that leads to long settling times. Table 12 summarizes the recommended segment length changes.

Table 12. Summary of Segment Lengths

Net/Signal Name	Segment	Original Length (in.)	Recommended Length (in.)
WRARR #	CC-CS#3	2.802	2.9
WRARR #	CC-CS#5	2.735	2.9
PA4	PP-CC	4.673	4.3
PA6	PP-CC	5.123	4.9
PA7	PP-CC	5.368	5.1
PA10	PP-CC	5.233	4.9
PA12	PP-CC	5.324	5.0
PA16	PP-CC	5.359	4.9

PP = Pentium processor
 CC = 82496 cache controller
 CS = 82491 cache SRAM

Actual system measurements have shown that the original segment lengths do not violate the specifications. The reduction in ringing is probably due to transmission line losses which are not accounted for in the simulation. Therefore for completed designs using the example layout these changes are not necessary; however, Intel does recommend that all future designs that use the layout example use these new lengths.

In addition, a layer change to the BRDYC1# routing is recommended. Section 7.5 Design Notes, describes that BRDYC1# was routed on an outer layer to reduce the propagation delay; however, this resulted in a signal quality violation. Since the original routing of the board, the flight time specification was relaxed and BRDYC1# can now be routed on an inner layer which allows it to meet both signal quality and flight time specifications.

7.6.11 I/O SIMULATION RESULTS FOR EACH NET

Electrical simulations were performed on each net within the optimized interface of the 256 Kbyte CPU-Cache Chip Set design example. The simulations were done at the fast and slow corners to verify that signal

Net	Fast	Slow
CD80	138.3	138.3
CD81	138.3	138.3
CD82	138.3	138.3
CD83	138.3	138.3
CD84	138.3	138.3
CD85	138.3	138.3

quality and flight time specifications are met. The simulations were done using TLC V4.1.13 from Quad Design Technology, Inc. using the files described in Section 7.6.2. Table 13 summarizes the simulation results assuming all the segment length changes listed in Section 7.6.10 have been implemented along with the layer change to the BRDYC1# routing.

Net	Fast	Slow
CD80	138.3	138.3
CD81	138.3	138.3
CD82	138.3	138.3
CD83	138.3	138.3
CD84	138.3	138.3
CD85	138.3	138.3

PP = Pentium processor
CC = 82489 cache controller
CS = 82481 cache SRAM

7.6.10 Pentium® PROCESSOR TO 82489 SEGMENT LENGTH AND ROUTING CHANGES

The example layout described in this application note was completed using early revisions to the I/O buffer models. This process was necessary to ensure that a board was available for the arrival of first silicon. After the model was improved based on the model values, the model was rechecked, the board layout was

rechecked. These simulations were run in the test environment to change the bus length between the Pentium processor and 82489 cache controller. These changes result in a better layout that meets the specifications. In particular, these changes reduce the amount of feedback and the timing that leads to long settling times. Table 13 summarizes the recommended segment length changes.

Table 13. Summary of Segment Lengths

Net/Signal Name	Segment	Original Length (in.)	Recommended Length (in.)
WRAP#	CC-CS#3	2.903	2.9
WRAP#	CC-CS#5	2.738	2.8
PA#	PP-CC	4.673	4.8
PA#	PP-CC	2.153	4.8
PA#	PP-CC	5.388	5.5
PA10	PP-CC	2.233	4.8
PA12	PP-CC	2.324	5.0
PA18	PP-CC	2.388	4.8

PP = Pentium processor
CC = 82489 cache controller
CS = 82481 cache SRAM

In addition, a layer change to the BRDYC1# routing is recommended. Section 7.2 Design Notes describes that BRDYC1# was routed on an outer layer to reduce the propagation delay; however, this resulted in a signal quality violation. Since the original routing of the board, the flight time specification was relaxed and BRDYC1# can now be routed on an inner layer which allows it to meet both signal quality and flight time specifications.

Actual system measurements have shown that the original segment lengths do not violate the specifications. The relaxation in timing is probably due to the timing model which was not accounted for in the simulation. Therefore, recommended designs using the example layout have changes to not necessary; however, Intel does recommend that all future designs that use the layout example use these new lengths.

Table 13. Summary of Simulation Results

Net	Flight Time (ns)	Signal Quality			
		Over/ Undershoot (V)	Ringback (V)	Settling Time (ns)	Time Beyond Supply (ns)
Specs.	Vary by Pin	3.0	1.75	12.5	6.0
82496 Driving					
A3-16 at CPU	7.2	2.1	1.0	16.5	7.4
A3-16 at SRAM	7.0	2.1	1.0	16.5	7.4
A17-31 at CPU	2.6	3.0	1.75	10.3	3.6
BT0-3	2.6	3.0	1.75	10.3	3.6
AHOLD	1.1	3.0	1.75	9.0	2.1
AP	1.4	3.0	1.75	9.0	2.1
BRDYC1#	0.9	2.9	1.7	8.3	1.8
EADS#	0.9	2.9	1.7	7.8	1.8
EWBE#	0.7	2.5	1.4	6.1	1.5
INV	1.6	2.8	1.5	12.4	2.9
KEN#	1.0	2.9	1.75	8.5	2.0
NA#	0.9	2.9	1.7	7.9	1.9
WB/WT#	0.9	2.8	1.7	7.5	1.8
BLAST#	2.5	2.2	1.0	6.2	3.3
BLEC#	2.2	2.1	0.9	5.9	3.1
BOFF# at CPU	2.5	2.6	1.1	12.1	3.0
BOFF# at SRAM	2.9	2.6	1.1	12.1	3.0
BRDYC2#	2.5	2.2	1.1	6.2	3.7

3

Table 13. Summary of Simulation Results (Continued)

Net	Flight Time (ns)	Signal Quality			
		Over/ Undershoot (V)	Ringback (V)	Settling Time (ns)	Time Beyond Supply (ns)
BUS#	2.5	2.1	0.9	6.5	3.3
MAWEA#	2.6	2.2	1.0	8.1	3.4
MCYC#	2.6	2.1	0.8	6.5	3.3
WAY	2.5	2.1	0.9	6.4	3.6
WBA	2.7	2.2	1.0	6.7	3.5
WB TYP	2.6	2.2	1.0	6.1	3.6
WBWE#	2.6	2.1	0.9	6.2	3.3
WRARR#	2.5	2.4	1.1	8.1	3.3
Pentium™ Processor Driving					
A3-16 at Controller	2.5	2.6	1.2	12.8	3.2
A3-16 at SRAM	2.8	2.6	1.2	12.8	3.2
A17-31	1.5	3.0	1.8	10.6	2.3
BT0-3	1.5	3.0	1.8	10.6	2.3
D0-63, DP0-7	1.2 min. 1.4 max.	3.0	1.8	11.5	2.5
ADS#	2.6	2.2	1.0	7.3	3.7
HITM# at Controller	3.0	3.2	1.6	20.3	4.1
HITM# at SRAM	3.2	3.2	1.6	20.3	4.1
W/R# at Controller	3.0	3.1	1.6	20.7	4.1
W/R# at SRAM	3.2	3.1	1.6	20.7	4.1
ADSC#	1.2	3.0	1.75	7.3	1.7
AP	1.3	3.0	1.75	8.8	2.1
CACHE#	1.1	2.9	1.75	7.2	1.7
D/C#	1.1	2.9	1.75	7.1	1.7
LOCK#	1.3	3.0	1.8	8.6	2.0
M/IO#	1.4	3.0	1.8	9.1	2.1

Table 13. Summary of Simulation Results (Continued)

Net	Flight Time (ns)	Signal Quality			
		Over/Undershoot (V)	Ringback (V)	Settling Time (ns)	Time Beyond Supply (ns)
PCD	1.1	2.9	1.7	6.9	1.6
PWT	1.2	3.0	1.8	8.7	1.9
SCYC	1.2	3.0	1.75	7.4	1.7
BE0-7 #	1.9	3.5	2.0	14.7	3.3
82491 Driving					
D0-63, DP0-7	1.7 min. 1.9 max.	3.0	1.9	12.1	2.7

Shading indicates a spec. violation.

The shaded entries indicate specification violations with the example design layout. Intel is continuing to work on addressing these violations by either relaxing specifications or improving the layout design. Note that no violations have been measured on the actual board. The violations have all been in simulation.

7.7 Possible Modification to the Layout

7.7.1 NON-PARITY LAYOUT

Intel has not simulated a non-parity layout example. The following suggestions will assist in modifying the design example for non-parity implementations. You must simulate all paths that are altered when the parity components are removed to ensure that flight time and signal quality specifications are still met.

Modify the following aspects of the layout example:

1. Remove the two leftmost 82491 components, U9 and U10. These are the parity components.
2. Rework Topologies 1 and 1b. Balance the array so that the two critical branch points branch out to electrically equivalent traces, i.e., adjust L_w to be electrically equivalent to $L_v + L_z$. Keep the trace leading to the Pentium processor length L_a . Topologies 1NP and 1bNP illustrate this (NP = Non-Parity). Also, retune L_b to be electrically equivalent with these new trace lengths. Topologies 1 and 1b indicate exactly where the critical branch points are.
3. Rework topologies 3 and 3b. Make the four traces branching from the 82496 electrically equivalent. This may be accomplished by making $L_g = L_h$ for these topologies.

4. Remove the Byte Enable traces that connect to the parity chips.

Making traces electrically equivalent means that reflections from all branches return to the source at the same point in time. In simple cases, electrically equivalent traces are the same length. In all cases, simulate the effects of changing trace lengths to find the proper trace length and routing.

8.0 512K CPU-CACHE CHIP SET OPTIMIZED INTERFACE LAYOUT DESIGN EXAMPLE

This chapter contains an example layout design for Intel's 512 Kbyte CPU-Cache Chip Set's optimized interface. Intel has simulated and verified the example layout using the latest information. Work is currently underway to validate the design by measuring the flight time and signal quality parameters on boards based on the design example. As updated information becomes available on the components and the boards, Intel plans to update this example accordingly.

The intent of the design example is to provide system designers a starting point. It provides one solution of how the Pentium processor, 82496, and 82491 components can be placed and routed to ensure flight time and signal quality specifications are met. It is not the only solution. System designers can alter the layout to meet their system requirements as long as the flight time and signal quality specifications are met.

8.1 Layout Objectives

The 512K layout is an example of a CPU-Cache chip set arrangement that meets Intel's chip set specifications. The layout consists of 1 Pentium processor, 1 82496 cache controller, and 18 82491 cache SRAMs for a 512K second-level cache with parity. Although the layout is specifically designed for a chip set with parity, we will also discuss conversion to a non-parity layout.

This example layout targets the chip set's flight time and signal quality specifications. In addition to meeting those specifications, we had the following objectives:

1. To design the optimized portion of the interface so that the layout is not limited by interconnect performance. By not artificially creating any critical paths, the interface can yield maximum performance of the chip set.
2. To be consistent with EMI and thermal requirements.

3. To have the layout be used as a validation and correction vehicle by Intel. Intel will use the layout to validate the optimized interface of the chip set, measure flight times and signal quality, and tune input and output buffers.

Provided are complete specifications for a board layout: part lists, board layer plots, and the electronic files in Gerber format. Also provided are a set of topologies and line lengths so it will be easy to understand how the layout was generated.

8.2 Component Placement

To meet flight time with clock skew restrictions we placed the parts in relative proximity to each other. At the same time, we ensured that the layout's Memory Bus Controller (MBC) interface signals are routable. Figure 59 illustrates how the chip set components are placed in the layout example. The dot indicates the location of pin 1. Figure 59 shows a component side view of the layout.

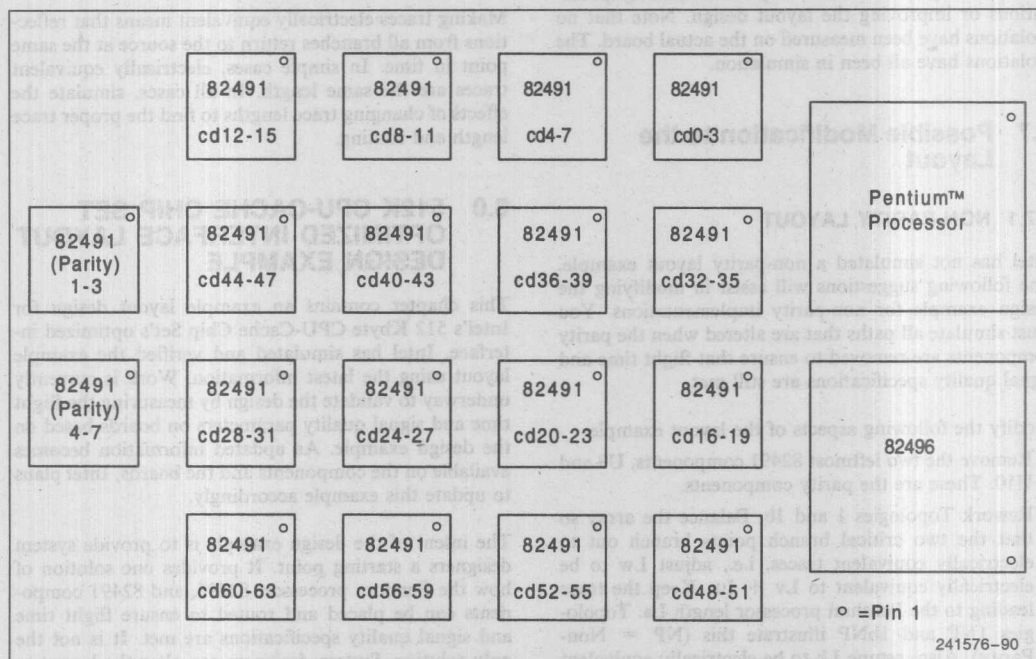


Figure 59. Component Placement

8.3 Signal Routing/Topologies

Table 14 and Table 15 list the signal nets and their corresponding topologies for the optimized and external interfaces of the CPU-Cache Chip Set.

All chip set signals in the optimized interface fall into six groups: low addresses, high addresses, Pentium processor control, 82496 control, CPU data, and byte enables. Within each group are subsets of signals that share common origination and destination points. Each subset has a unique routing called a "topology." Groups, subsets, and topologies are listed in Table 14.

Topologies are given only for signals that are routed to multiple chips. It is the system designer's responsibility for routing the "point-to-point" signals such as CADs#.

Topologies are also supplied for the external interface. These topologies provide channels for routing signals from the chip set components to the periphery where they can be connected to the memory bus and memory bus controller (MBC). However, topologies are not supplied for point to point signals in the MBC interface (e.g. CRDY#). Instead, the system designer must optimize these for the particular application.

Table 15 lists the topologies provided for the MBC interface signals which are not point to point.

Figures 60 through 77 are the topologies which are described in Table 14 and 15. A topology is a graphical representation how specific sets of signals are routed. A topology shows the components that share a specific signal and the relative lengths of the traces between components.

Table 14. Optimized Interface Signal Net/Topology Assignments

Grouping	Routing Requirements	Topology
Low Addresses		
(PA3-PA17)	Bused to all core components. Must be routed to optimize delay and signal quality at all points.	1
High Addresses		
(PA18-PA31, PBT0-PBT3)	Point to point links. Must be kept as short as possible.	6
Pentium™ Processor Control		
(HITM#, W/R#)	Same as low addresses.	1
(ADS#)		5
(ADSC#, AP, CACHE#, D/C#, LOCK#, M/IO#, PCD, PWT, SCYC)	Same as high addresses.	6
CC Control		
(BUS#, MAWEA#, WBWE#, WBTYP#, WBA, BLAST#)	Must be routed to optimize delay and signal quality at the CS.	3
(BLEC#)	Not routed to parity CSs.	4
(BOFF#)		1
(AHOLD, EADS#, KEN#, BRDYC1#, INV, EWBE#, NA#, WB/WT#)	Same as high addresses.	6
(BRDYC2#, WRARR#, MCYC#, WAY)	Routed differently.	15
CPU Data		
(CD0-CD63, CP0-CP7)	Point to point signals. Keep as short as possible. Keep within 3" of each other to minimize skew.	6
Byte Enables		
(CBE0#-CBE7#)		7

Signal	Topology
MDATA0-63, Parity0-7	8
BRDY0#, CLK0	9
CRDY#	10
RESETC	17
MBRDY#, MOCLK, MDOE#	11
BRDY1-3#	12
MEOC1-3#	18
CLK1-3	13
MFRZ#, MSEL#, MZBT#, MCLK	14
RESETCPU	16

Topology	Routing Requirements
Low Addresses	
1	Used to all core components. Must be routed to optimize delay and signal quality at all points.
High Addresses	
6	Point to point links. Must be kept as short as possible.
Pentium Processor Control	
1	Same as low addresses.
5	
6	Same as high addresses.
CC Control	
3	Must be routed to optimize delay and signal quality at the CC.
4	Not routed to parity CCs.
1	
6	Same as high addresses.
15	Routed differently.
CPU Data	
6	Point to point signals. Keep as short as possible. Keep within 3" of each other to minimize skew.
Byte Enables	
7	

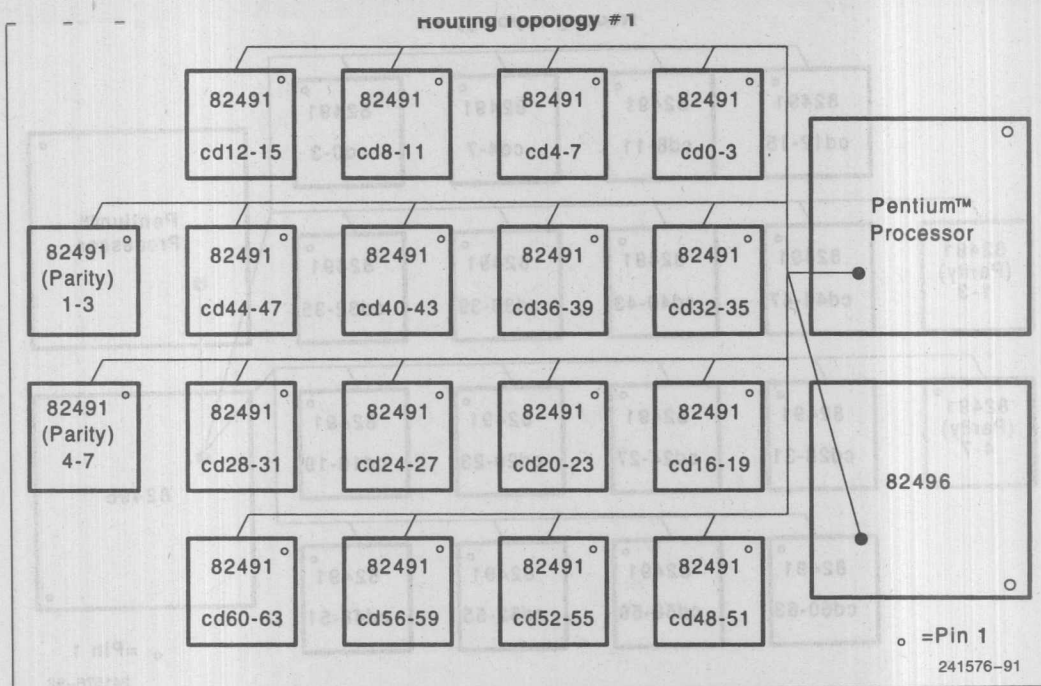


Figure 60. Topology 1

3

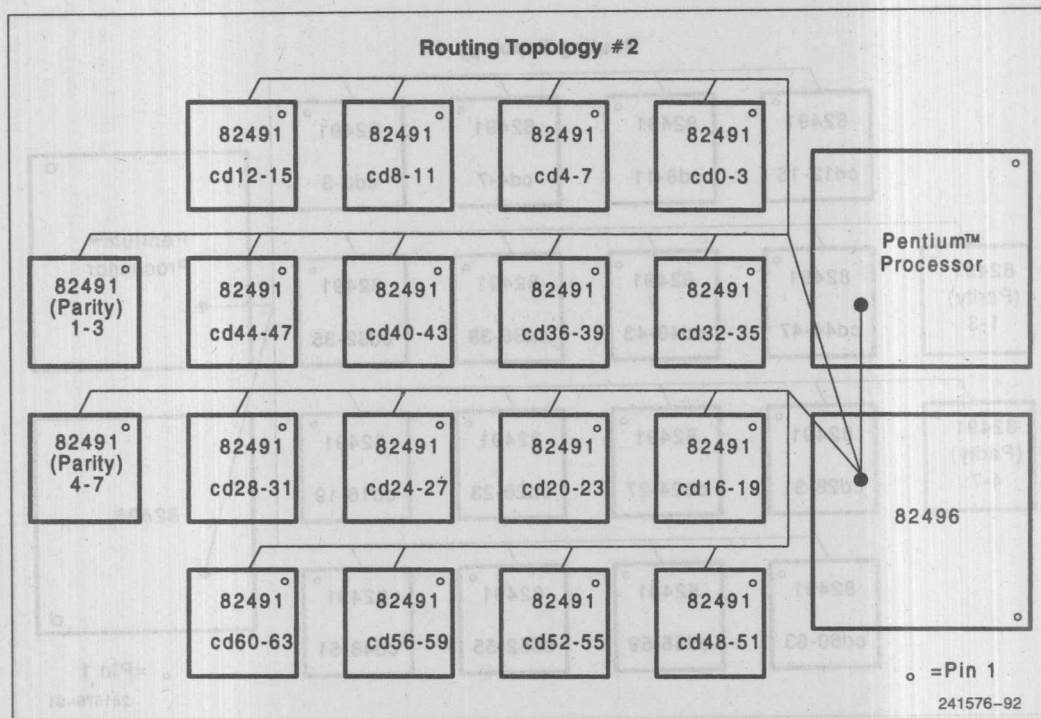


Figure 61. Topology 2

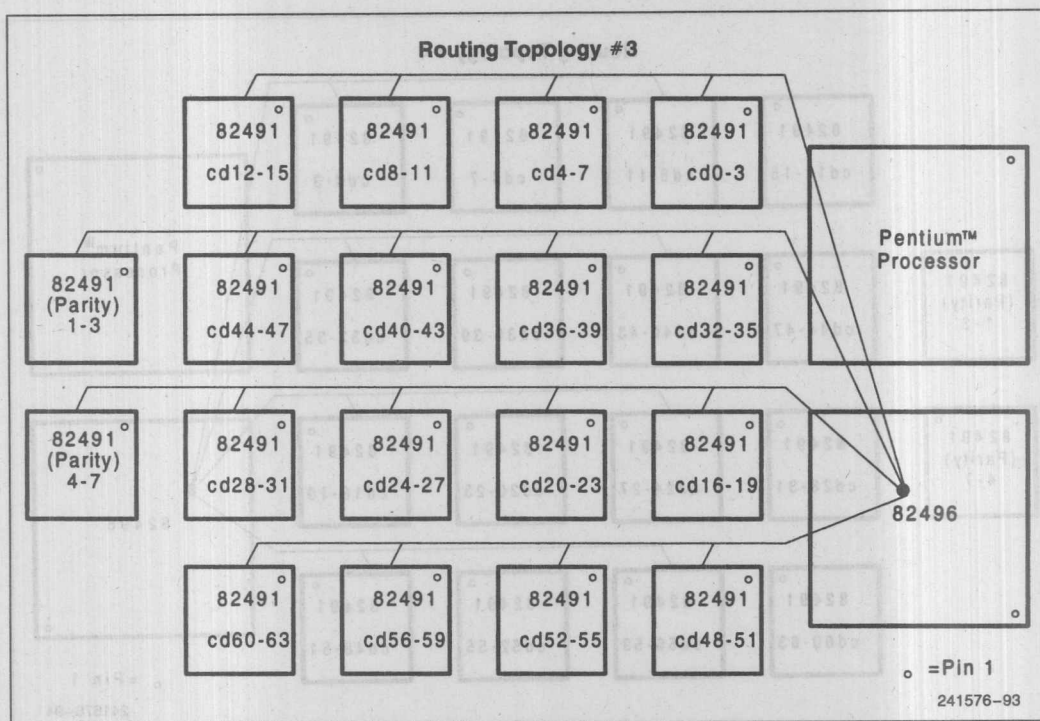


Figure 62. Topology 3

3

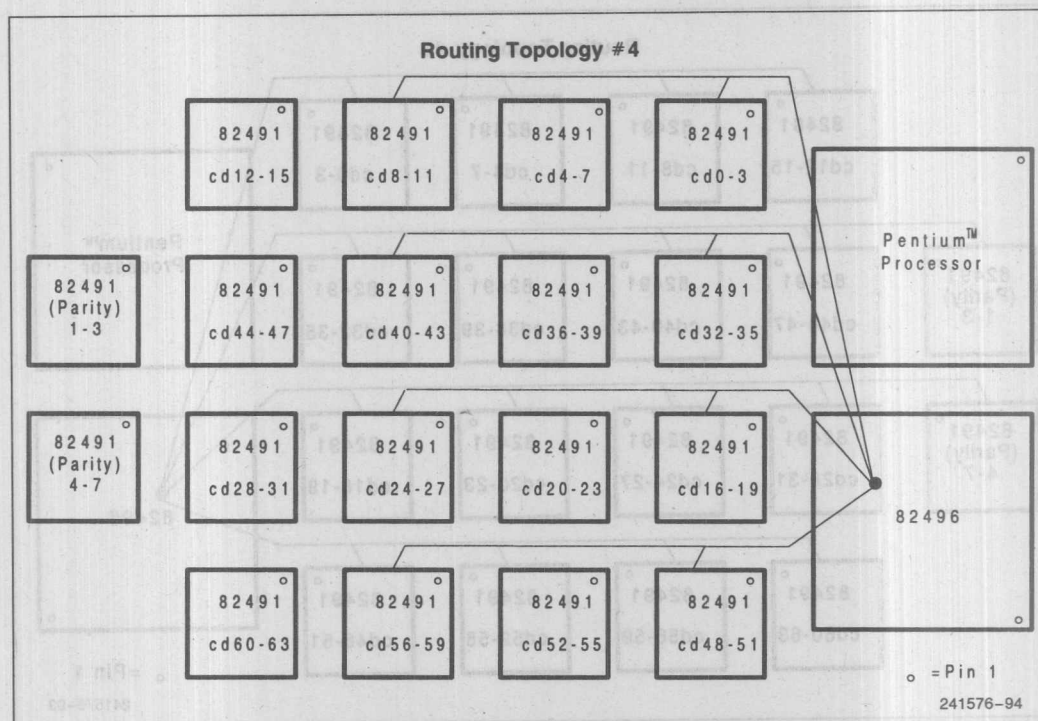


Figure 63. Topology 4

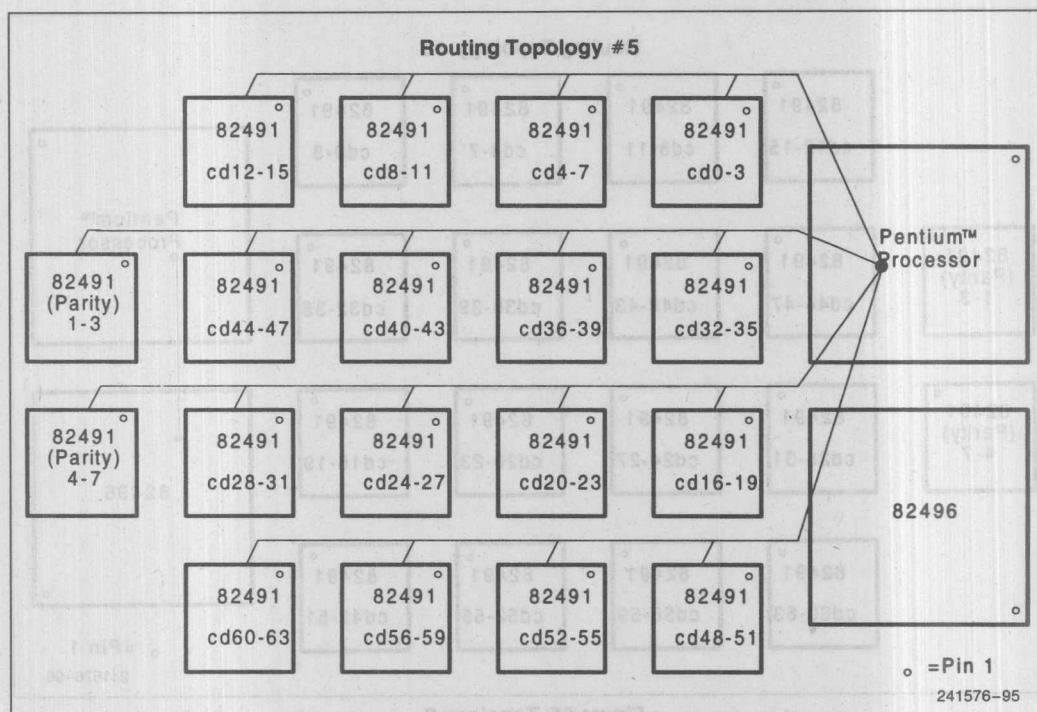


Figure 64. Topology 5

3

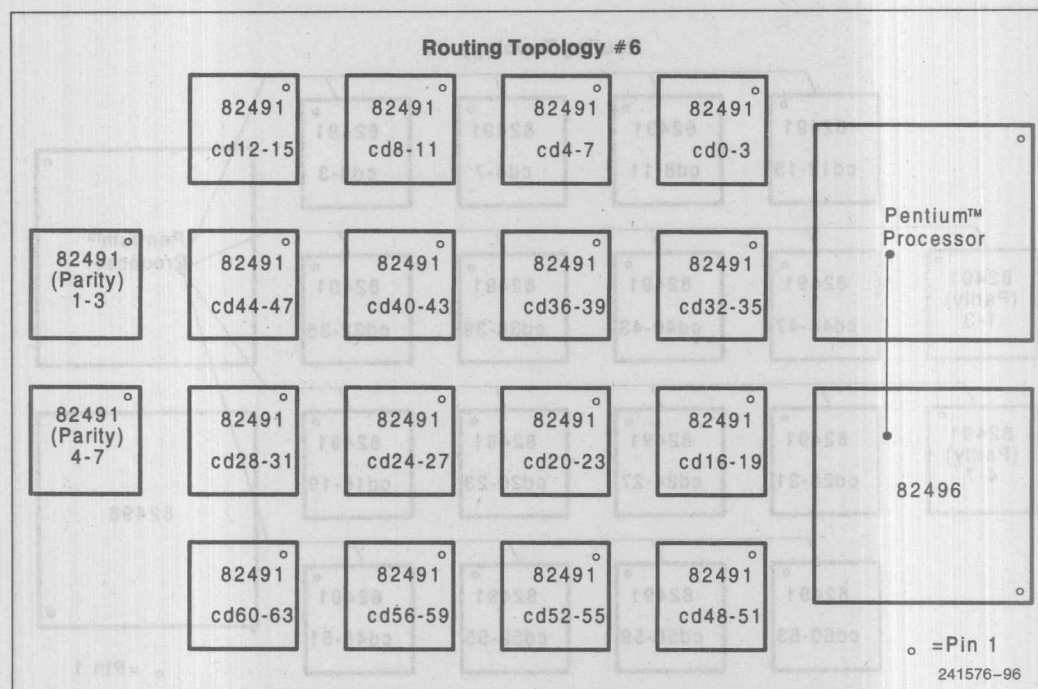


Figure 65. Topology 6

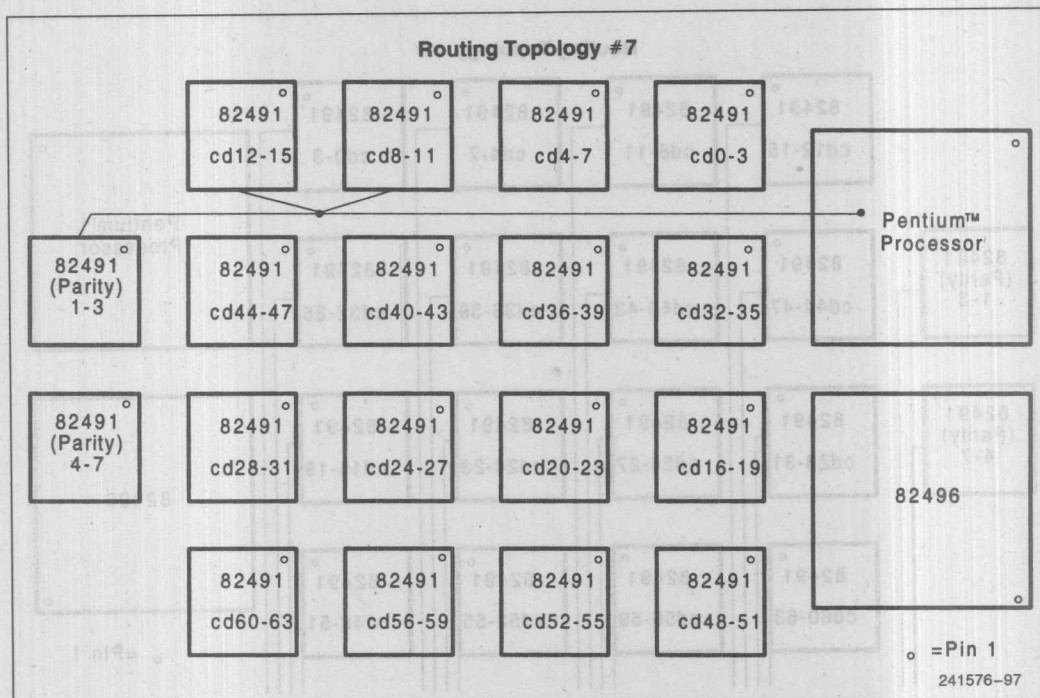


Figure 66. Topology 7

3

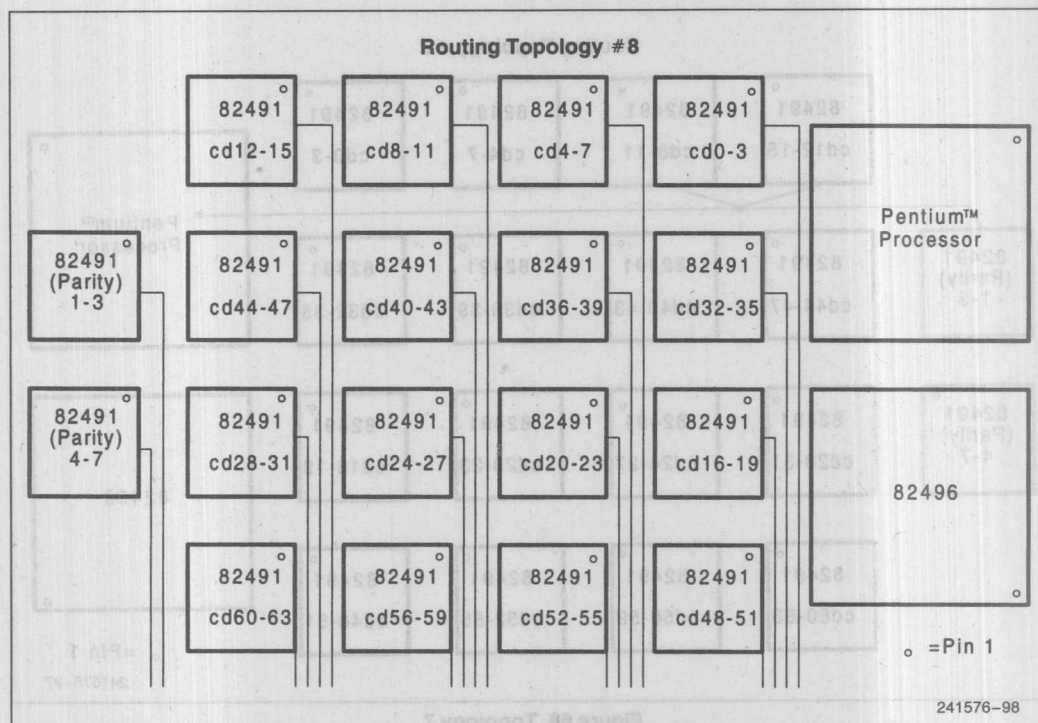


Figure 67. Topology 8

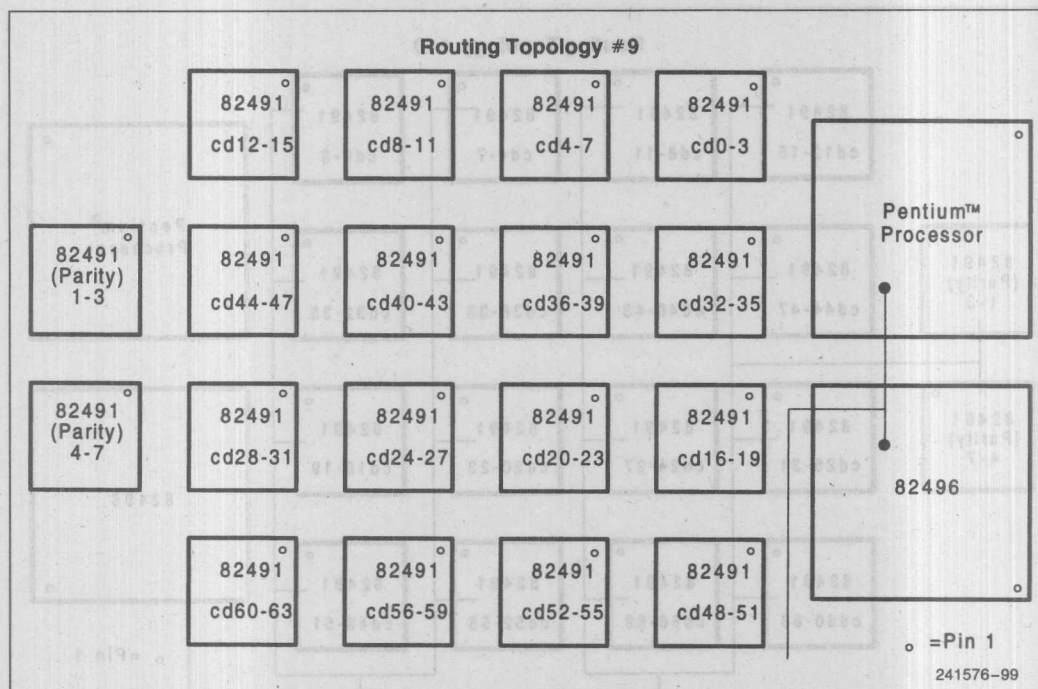


Figure 68. Topology 9

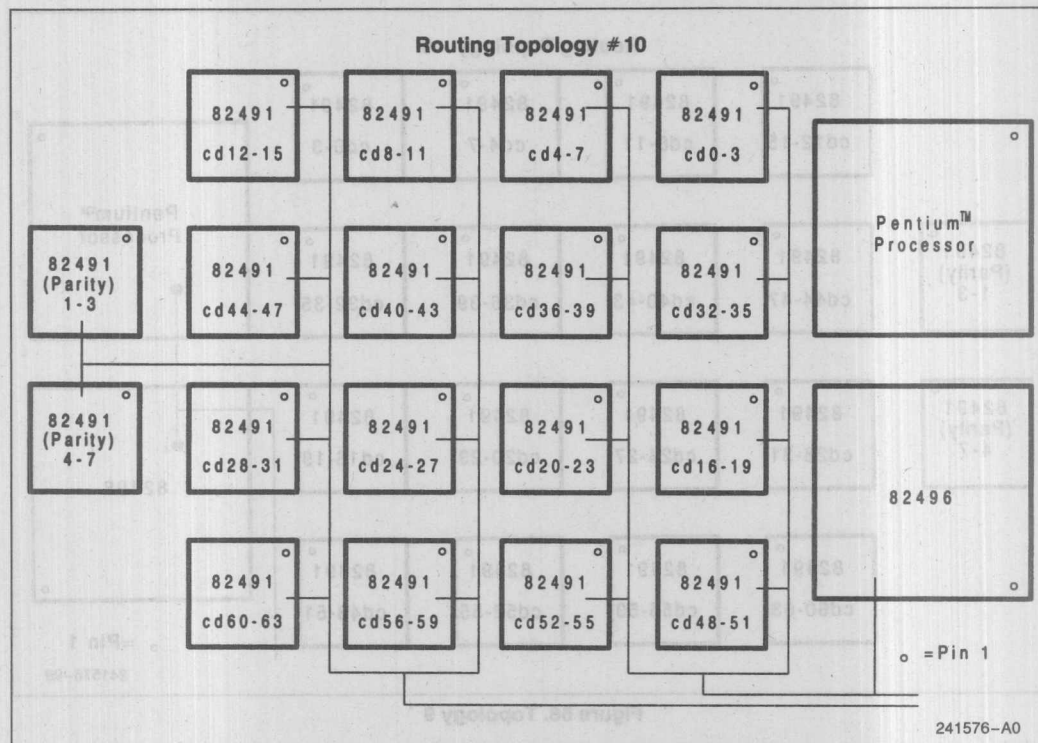


Figure 69. Topology 10

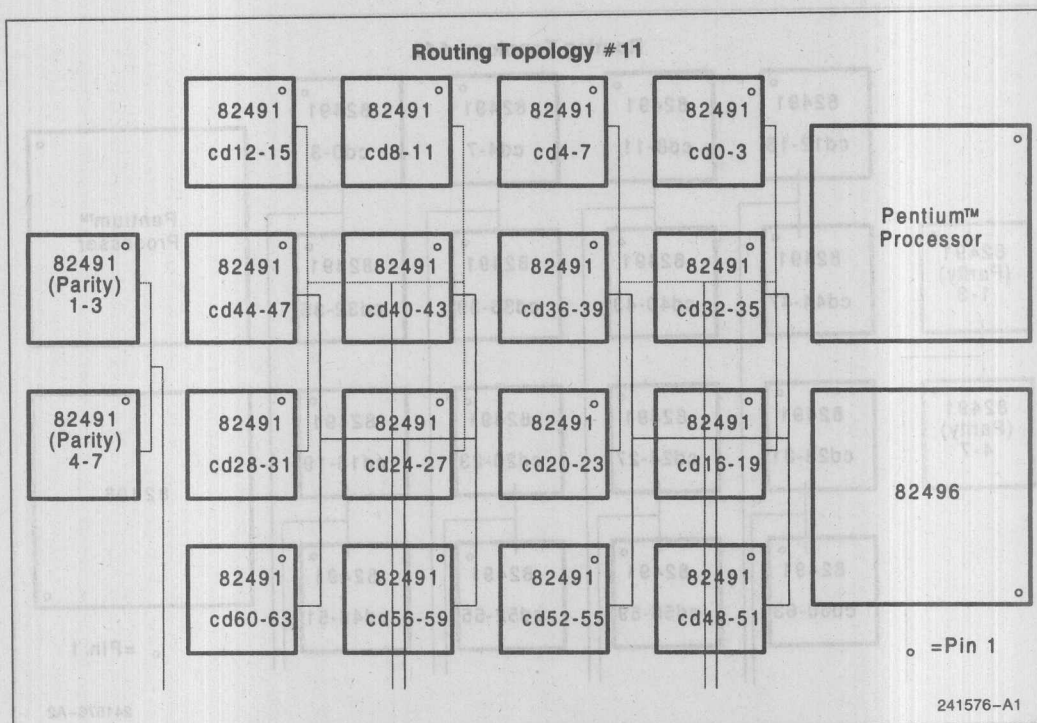


Figure 70. Topology 11

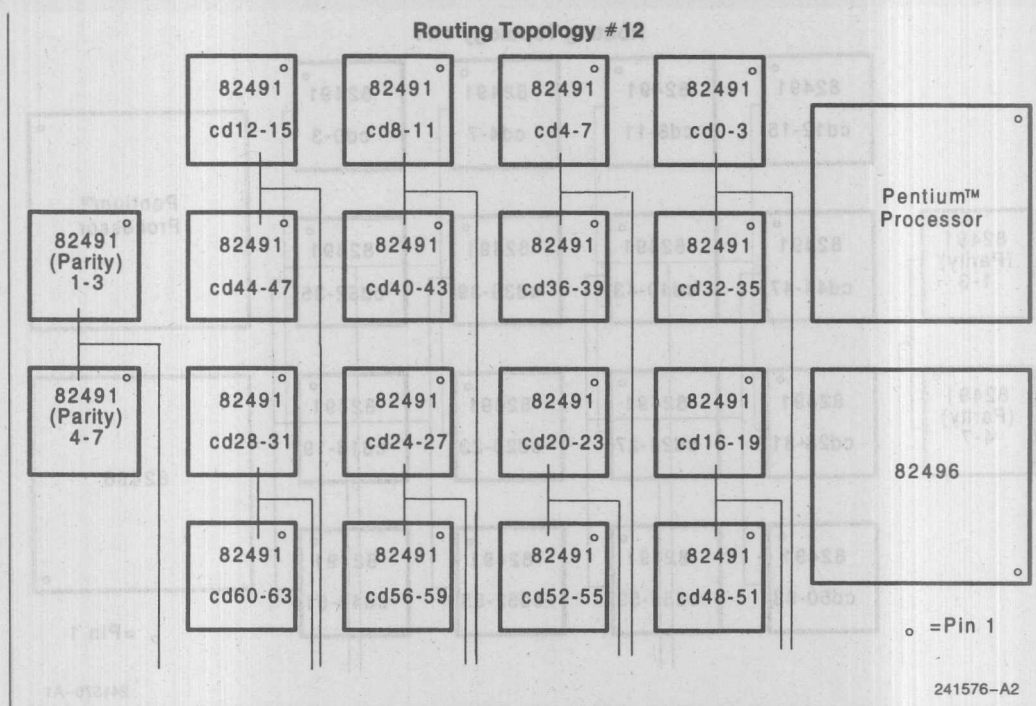


Figure 71. Topology 12

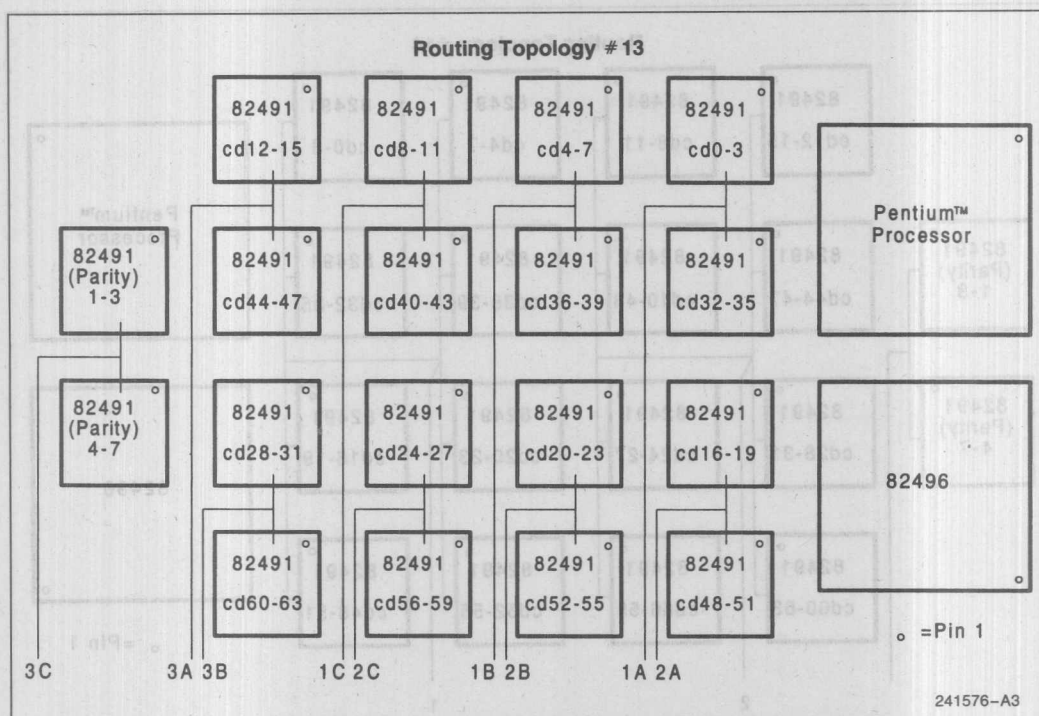


Figure 72. Topology 13

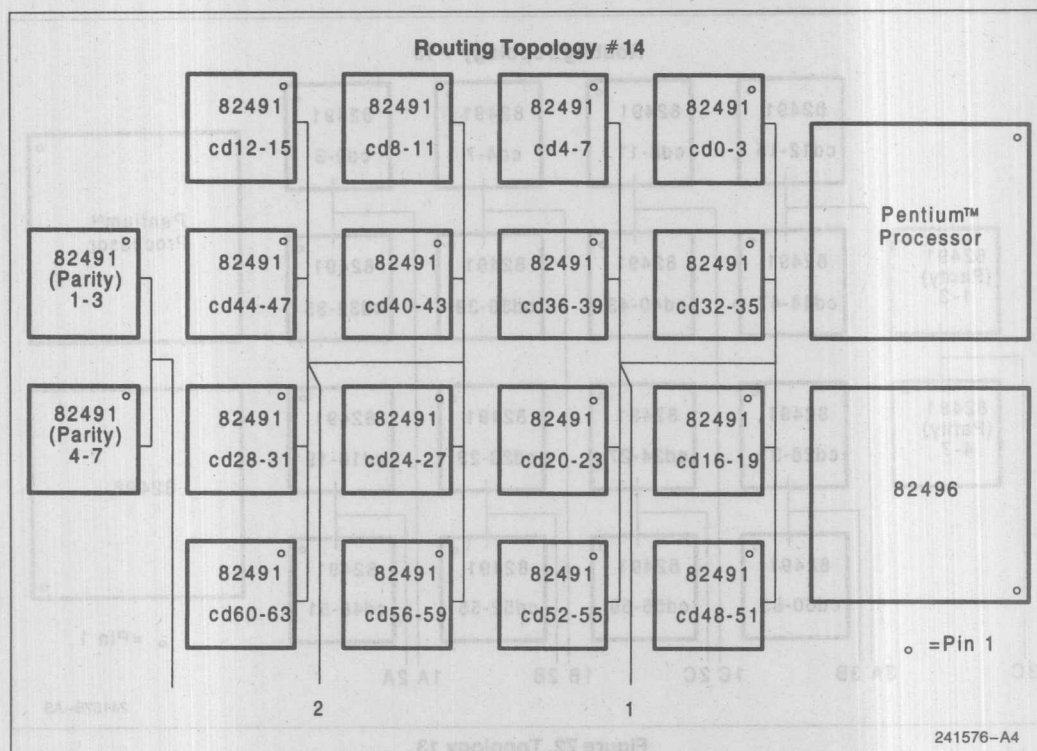


Figure 73. Topology 14

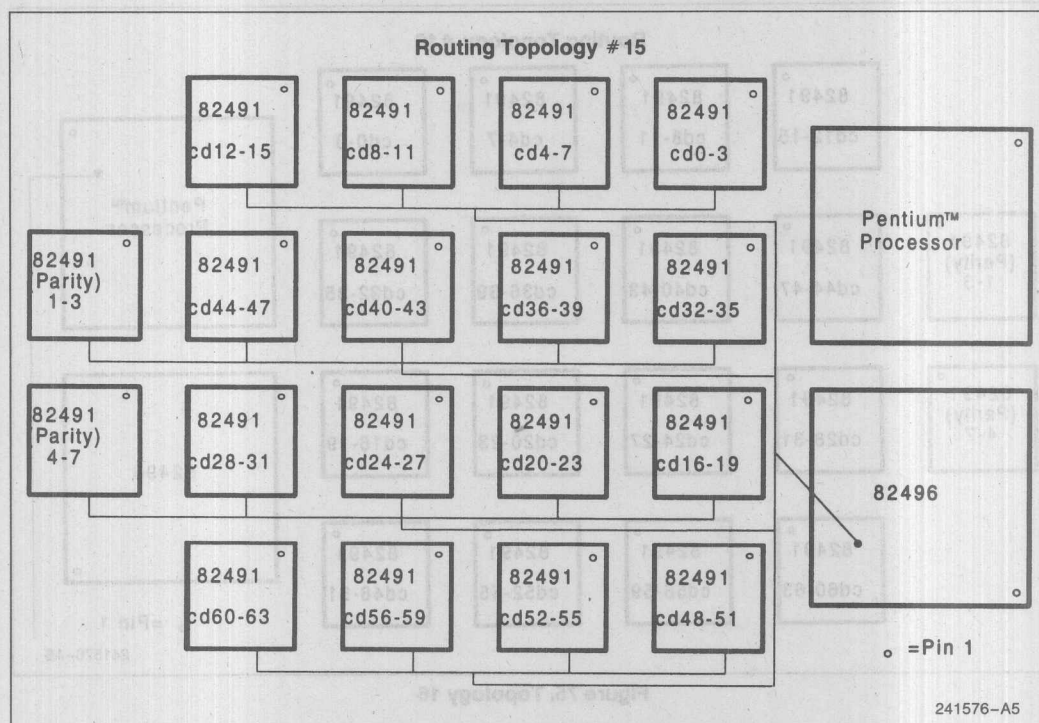


Figure 74. Topology 15

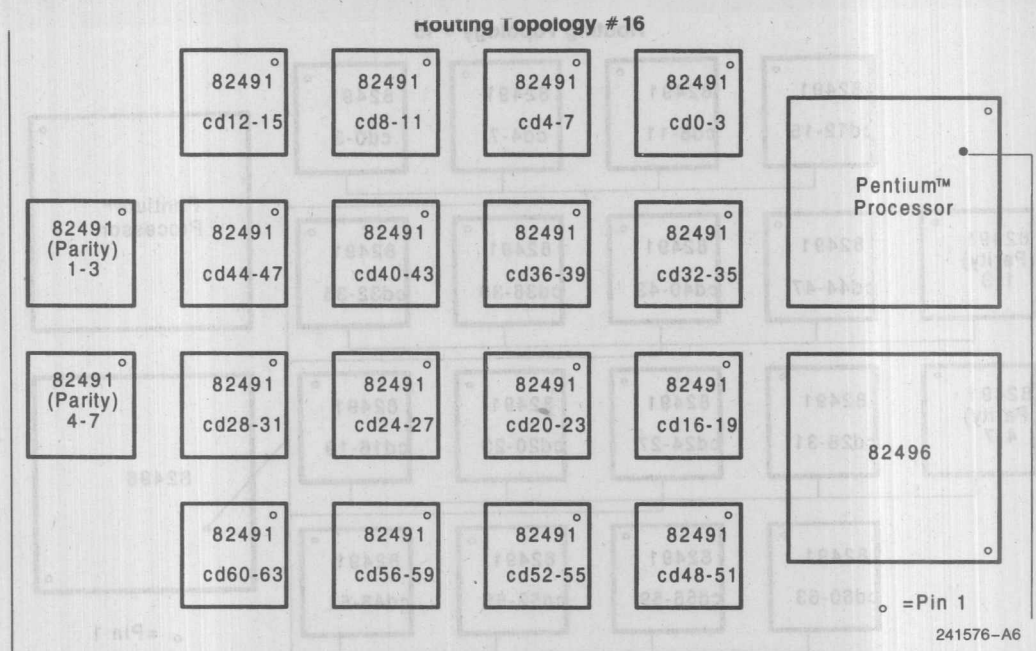


Figure 75. Topology 16

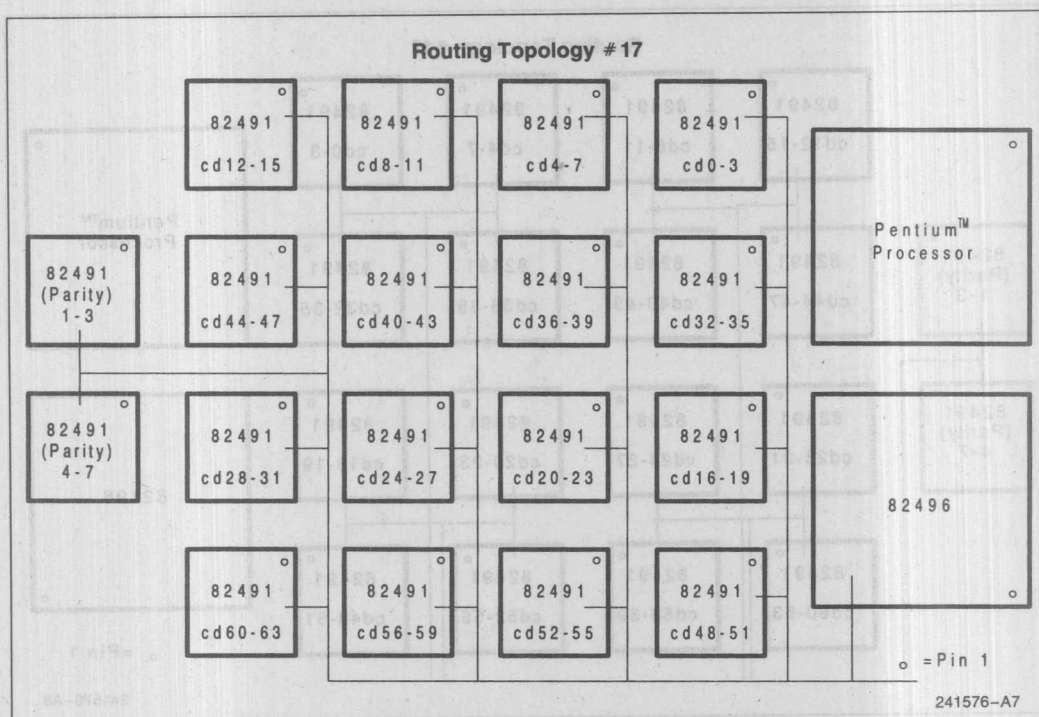


Figure 76. Topology 17

out. These properties were used as the specification for
guidelines the board manufacturer was to use in building
boards. Figure 75 provides the board layer stackup.

Specific board and trace properties were assumed while
performing the simulations to optimize the chip set lay-

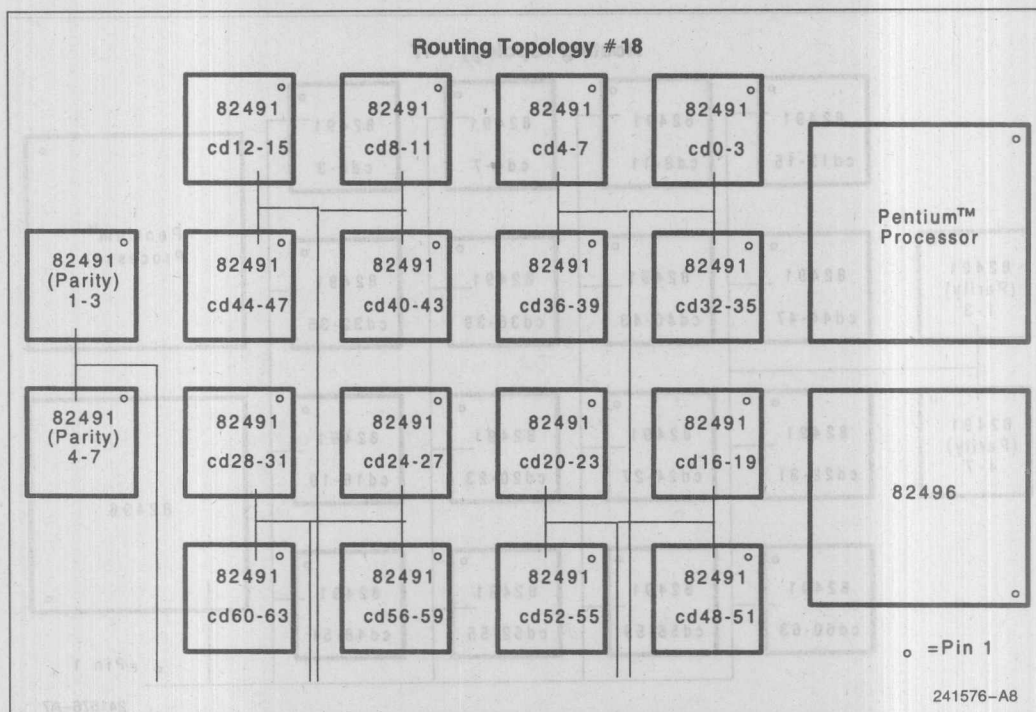


Figure 77. Topology 18

8.4 Board/Trace Properties

Specific board and trace properties were assumed while performing the simulations to optimize the chip set lay-

out. These properties were used as the specification or guideline the board manufacturer was to use in building boards. Figure 78 provides the board layer stackup.

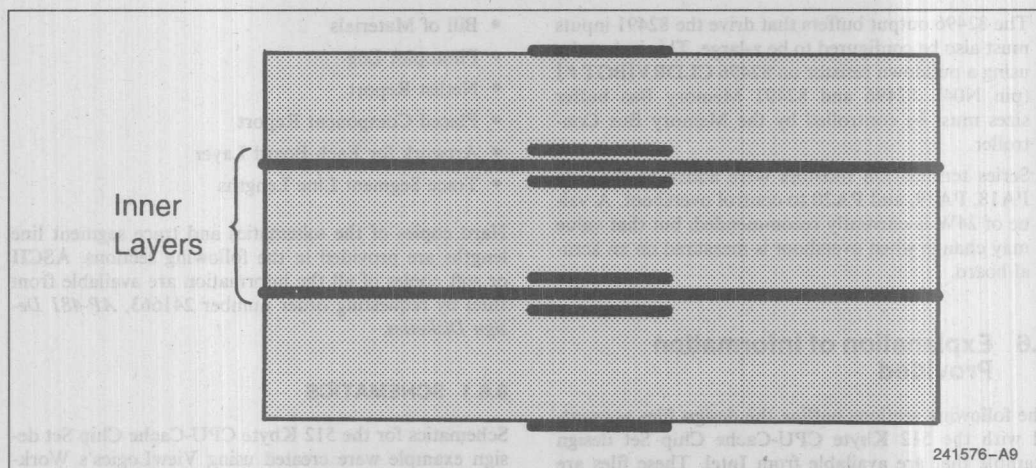


Figure 78. Board Layer Stackup

Table 16 lists the minimum and maximum trace characteristics. These parameters along with the board material determine the spacing between layers and the total board thickness. See Table 17.

Table 16. Trace Characteristics

	4 Inner Layers	2 Outer Layers
Width/Space	5/5 Mils	8/8.5 Mils
Z ₀	65W ± 10%	90W ± 20%
Velocity	1.85 to 2.41 ns/ft	1.35 to 2.05 ns/ft

Table 17. Other Printed Circuit Board Geometries

Via Pad	25 Mils
Via Hole	10 Mils
PGA Pad	55 Mils
PGA Hole	38 Mils
Layout Grid	5 Mils

Only the inner layers of the board are impedance controlled. The top and bottom layers are not impedance controlled.

8.5 Design Notes

The following design notes accompany this layout example:

1. The layout did not specifically address heat dissipation except to allow space for heat sinks to be attached. Please see the *Pentium™ Processor User's Manual* for the devices' thermal specifications. The *Pentium™ Processor Thermal Design Guidelines* application note provides some examples of possible thermal solutions.
2. All fast-switching signals are routed near the power and ground planes on inner layers of the board to minimize EMI effects. However, two sets of signals are routed on the top layer of the board: BRDYC1#, and JTAG signals. BRDYC1# is routed on top to take advantage of the higher trace velocity there. JTAG signals are routed on the top layer because they are low-speed signals and will probably be re-routed by each customer to suit individual needs.
3. Resistor R5 (0Ω) is used to set the Pentium processor configurable output buffers (A3–A20, ADS#, W/R#, and HITM#). When the resistor is included the buffers are set to the Extra Large size. When it is not included (BUSCHK# internally pulled high) the buffers are set to Large size. Intel currently recommends the x-large buffers be used for the 512K layout example. The 0Ω resistor should be designed into your design as Intel may change the recommended buffer size once silicon and the system design have been characterized.

4. The 82496 output buffers that drive the 82491 inputs must also be configured to be x-large. This is done by using a pulldown resistor on 82496 CLDRV[BGT#] (pin N04). 82496 and 82491 Memory Bus buffer sizes must be controlled by the Memory Bus Controller.
5. Series termination resistors were added to the nets PA18, PA19, and PA20 to control overshoot. A value of 24W is currently recommended, but that value may change when overshoot is measured on an actual board.

8.6 Explanation of Information Provided

The following sections outline the design files associated with the 512 Kbyte CPU-Cache Chip Set design example that are available from Intel. These files are provided to simplify the task of porting the design example into a specific design. By using these files, designers may eliminate or minimize the amount of duplicate effort when using the design example as the basis for their design. The following items are provided:

- Schematics
- I/O Model Files
- Board Files

- Bill of Materials
- Photoplot Log
- Netlist Report
- Placed Component Report
- Artwork for Each Board Layer
- Trace Segment Line Lengths

Hard copies of the schematics and trace segment line lengths are provided in the following sections. ASCII or soft copies of all the information are available from Intel by requesting order number 241663, *AP-481 Design Diskettes*.

8.6.1 SCHEMATICS

Schematics for the 512 Kbyte CPU-Cache Chip Set design example were created using ViewLogics's Workview V4.1. The schematics are 13 pages long. Both the Workview and the postscript files are available from Intel as described above.

Table 16: Trace Characteristics

Layer	Width/Space	Drill
1 Outer Layer	6/6.3 Mils	0.025
2 Inner Layer	6/6.3 Mils	0.025
3 Core	6/6.3 Mils	0.025
4 Prepreg	6/6.3 Mils	0.025

Table 17: Other Printed Circuit Board Geometries

Layer	Width	Drill
1 Outer Layer	6 Mils	0.025
2 Inner Layer	6 Mils	0.025
3 Core	6 Mils	0.025
4 Prepreg	6 Mils	0.025
5 Vias	6 Mils	0.025

PENTIUM™ PROCESSOR/82496/82491 512KB MODULE
7-1-92
REV 2.0

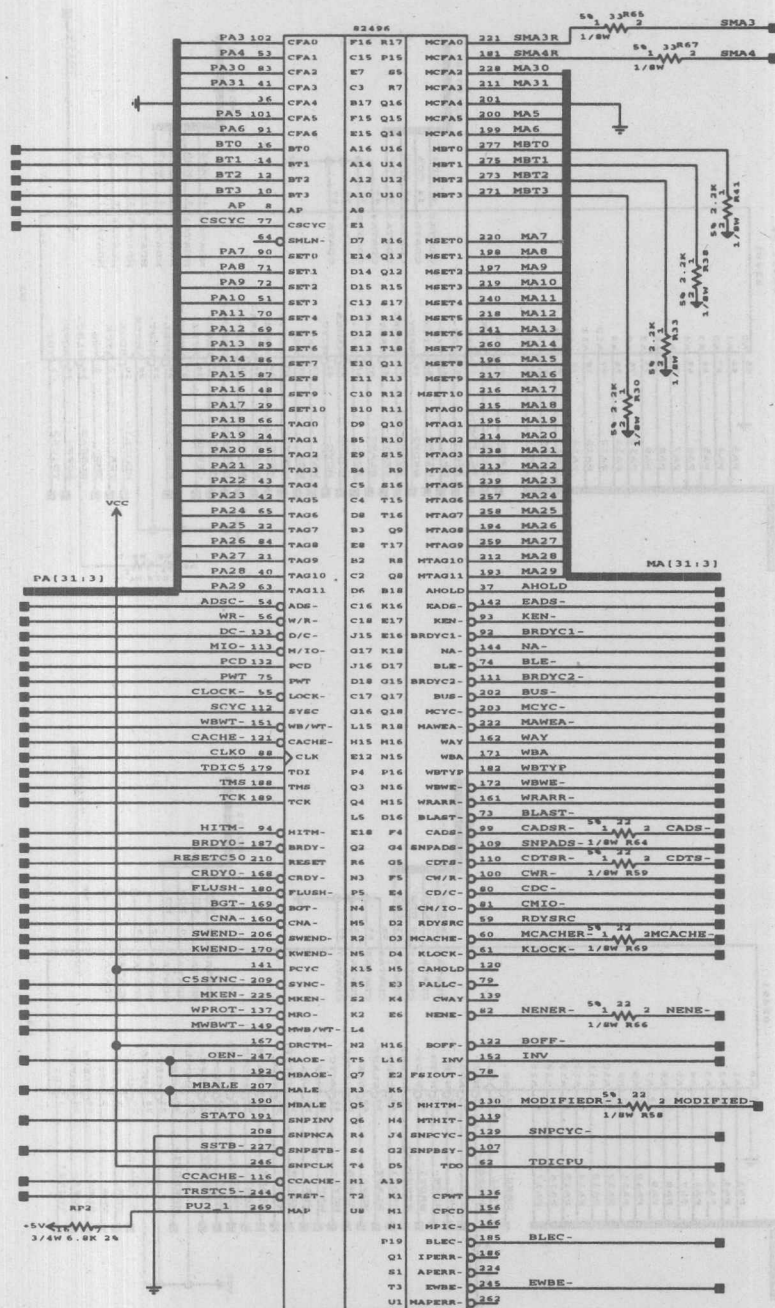
(Tied MBZT- on all 82491 to Vcc)
(Removed series resistor on clk7 and added dummy load on clk7 at crdy pal, pin 27)

NOTES:

(Unless Otherwise Specified)

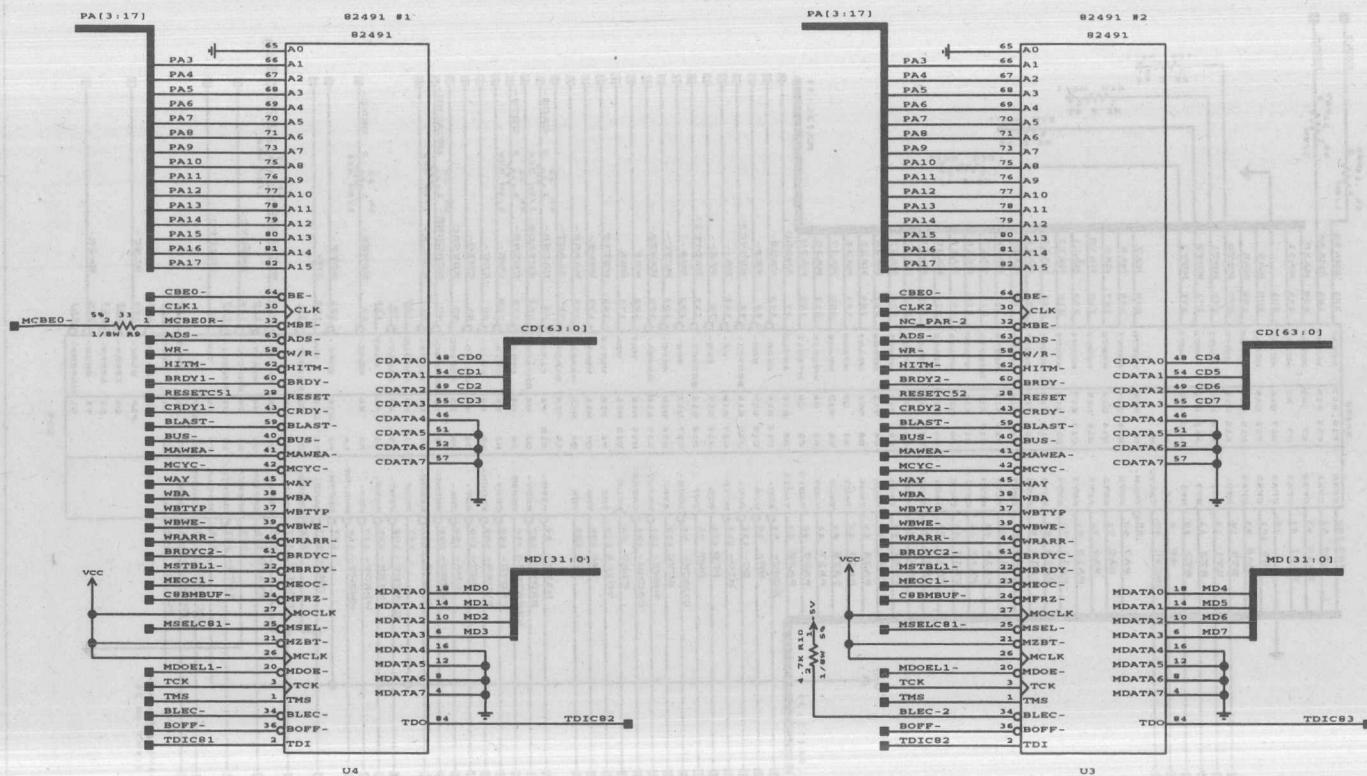
1. Capacitor values are in microfarads.
2. Resistor values are in ohms.
3. An "-" following a signal name denotes negation.
4. VCC = +5V.
5. This document also exists on electronic media.



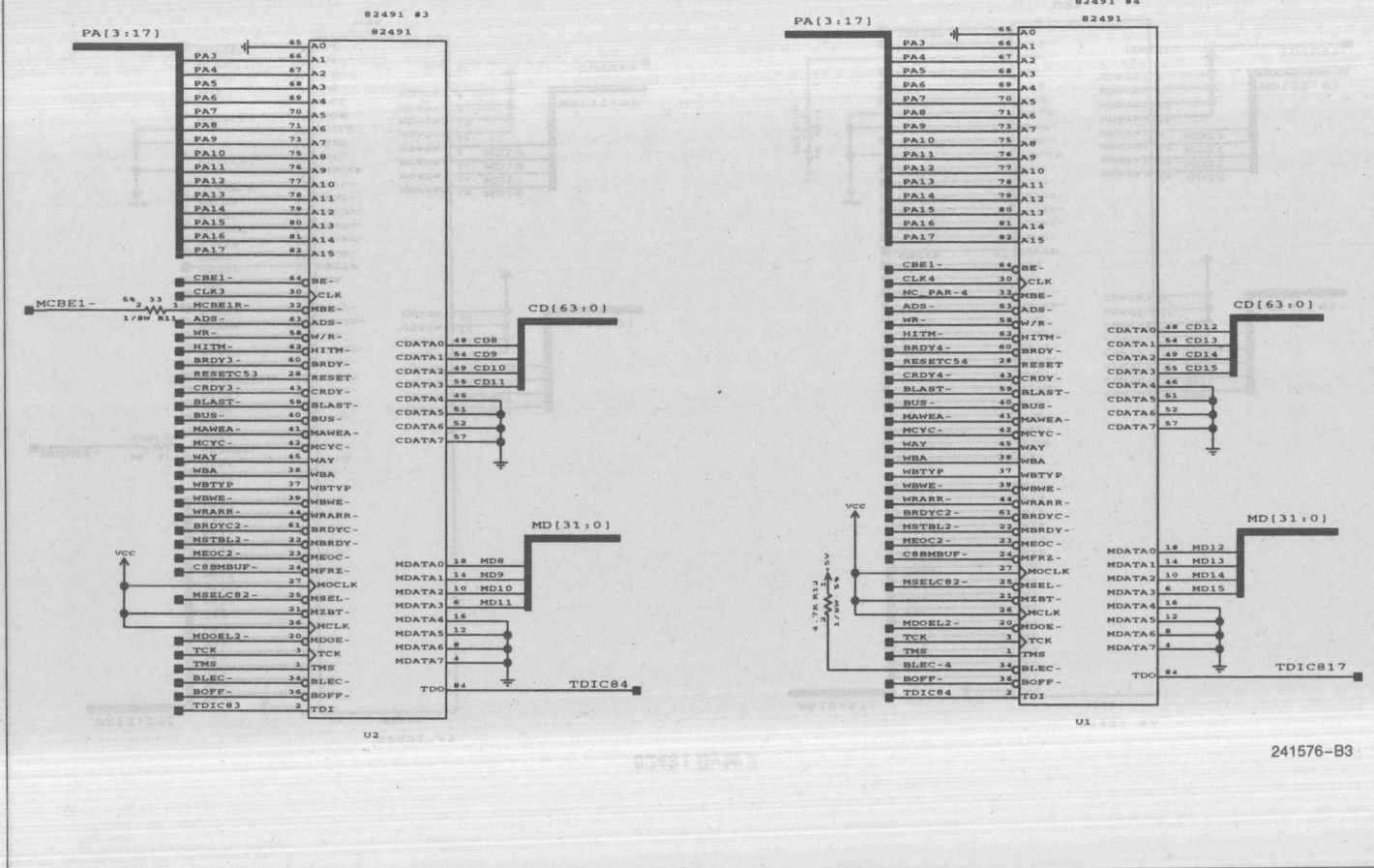


3

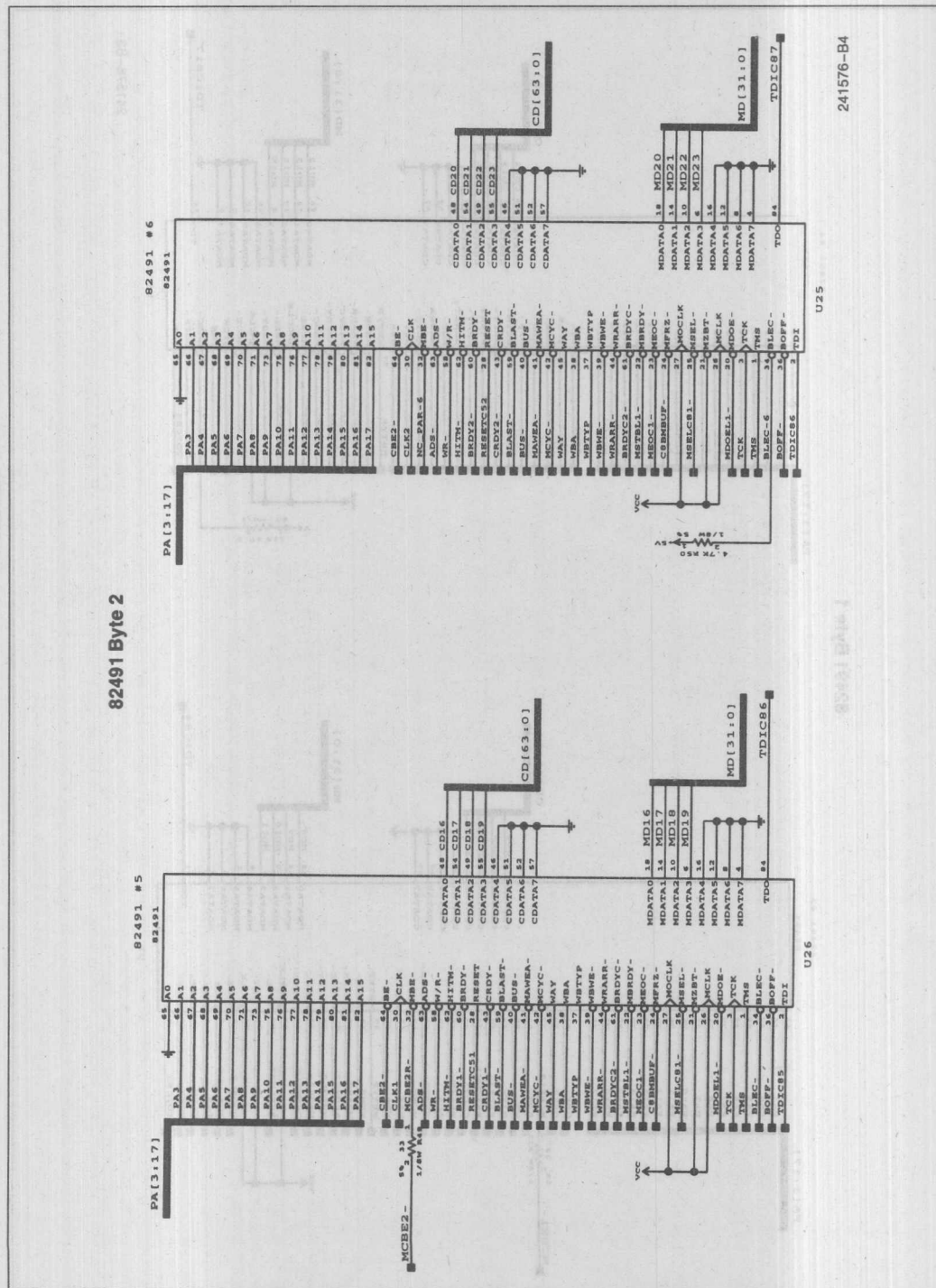
82491 Byte 0



82491 Byte 1



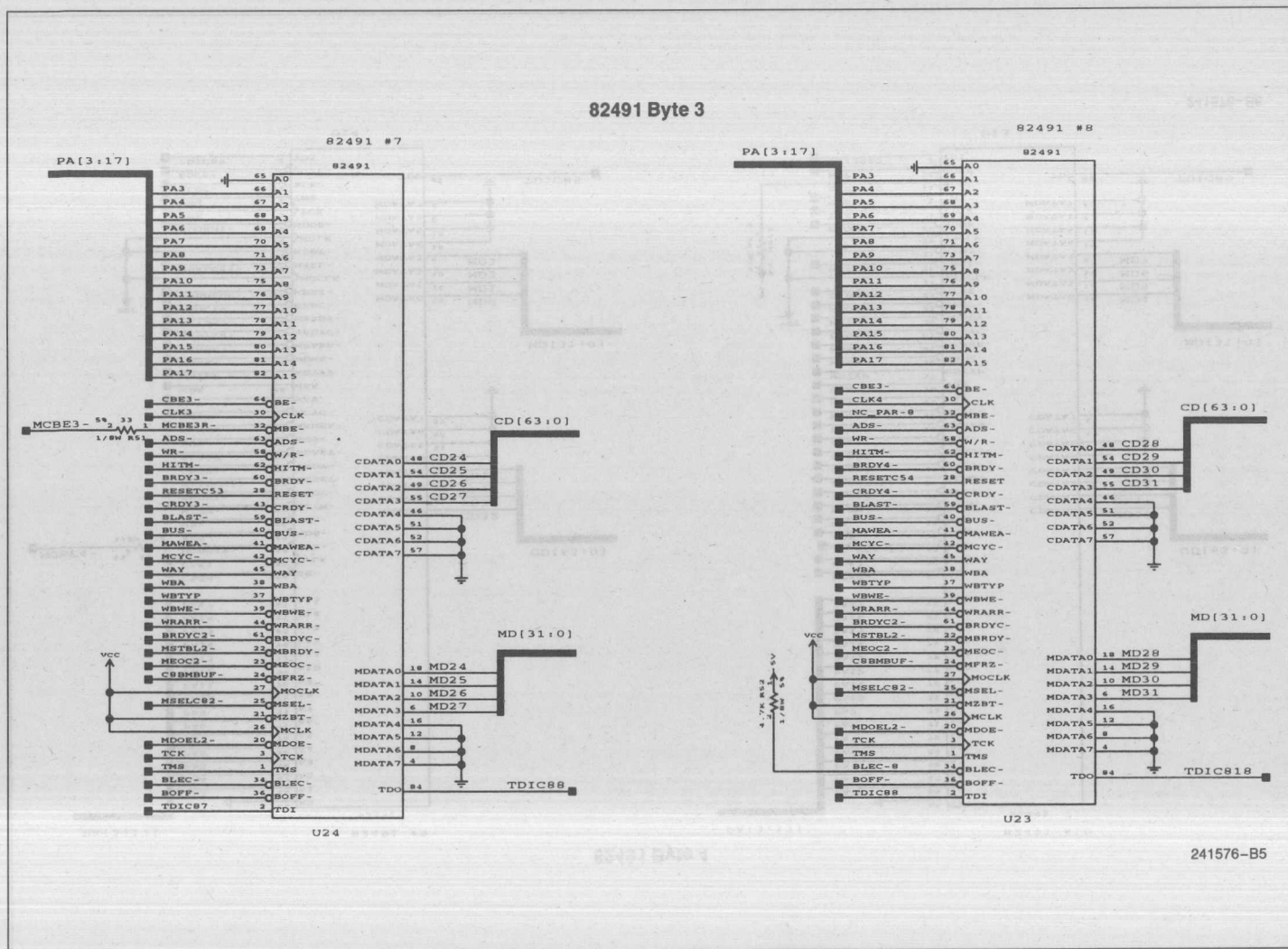
82491 Byte 2



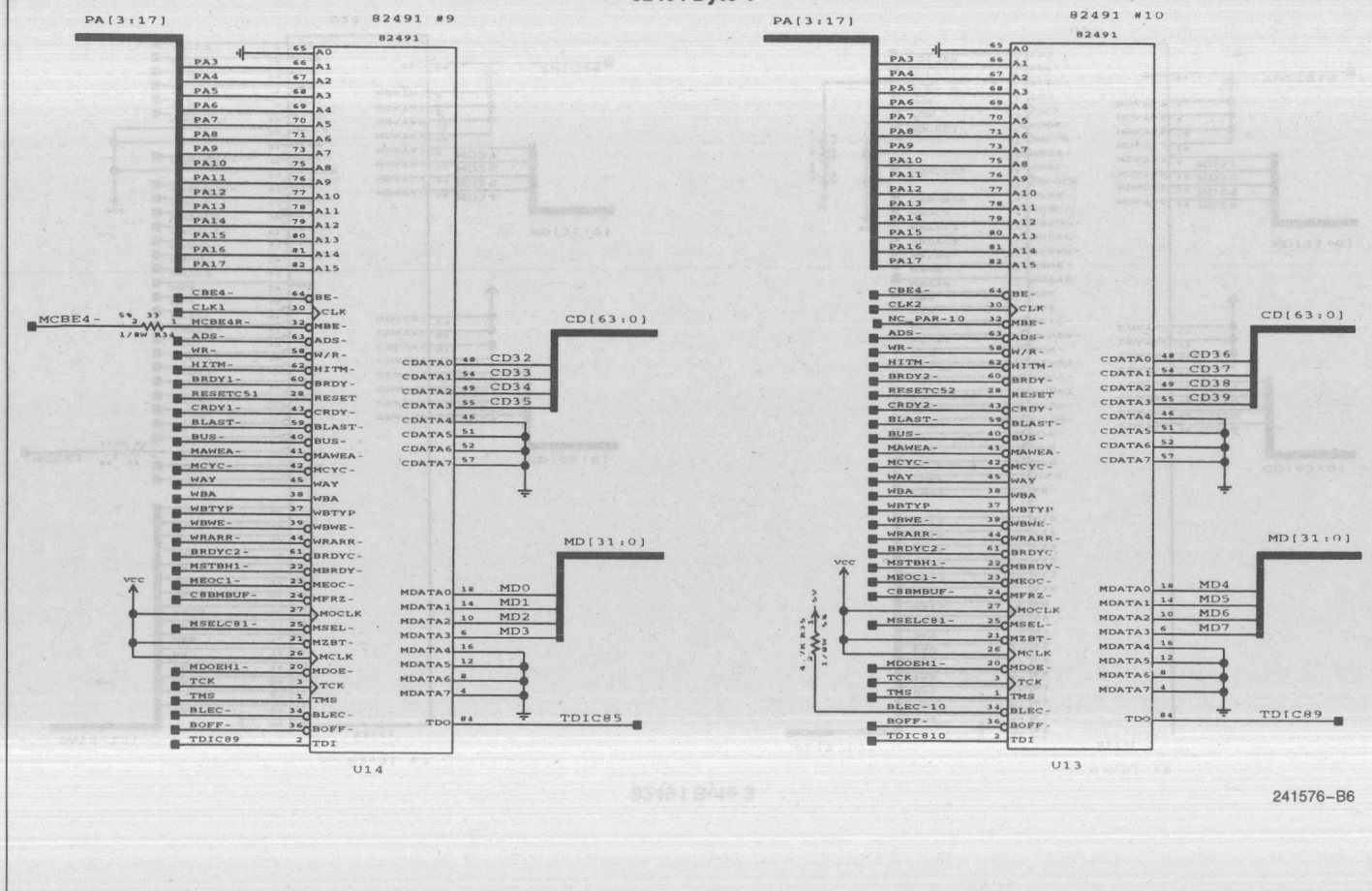
82491 Byte 3

intel®

AP-481

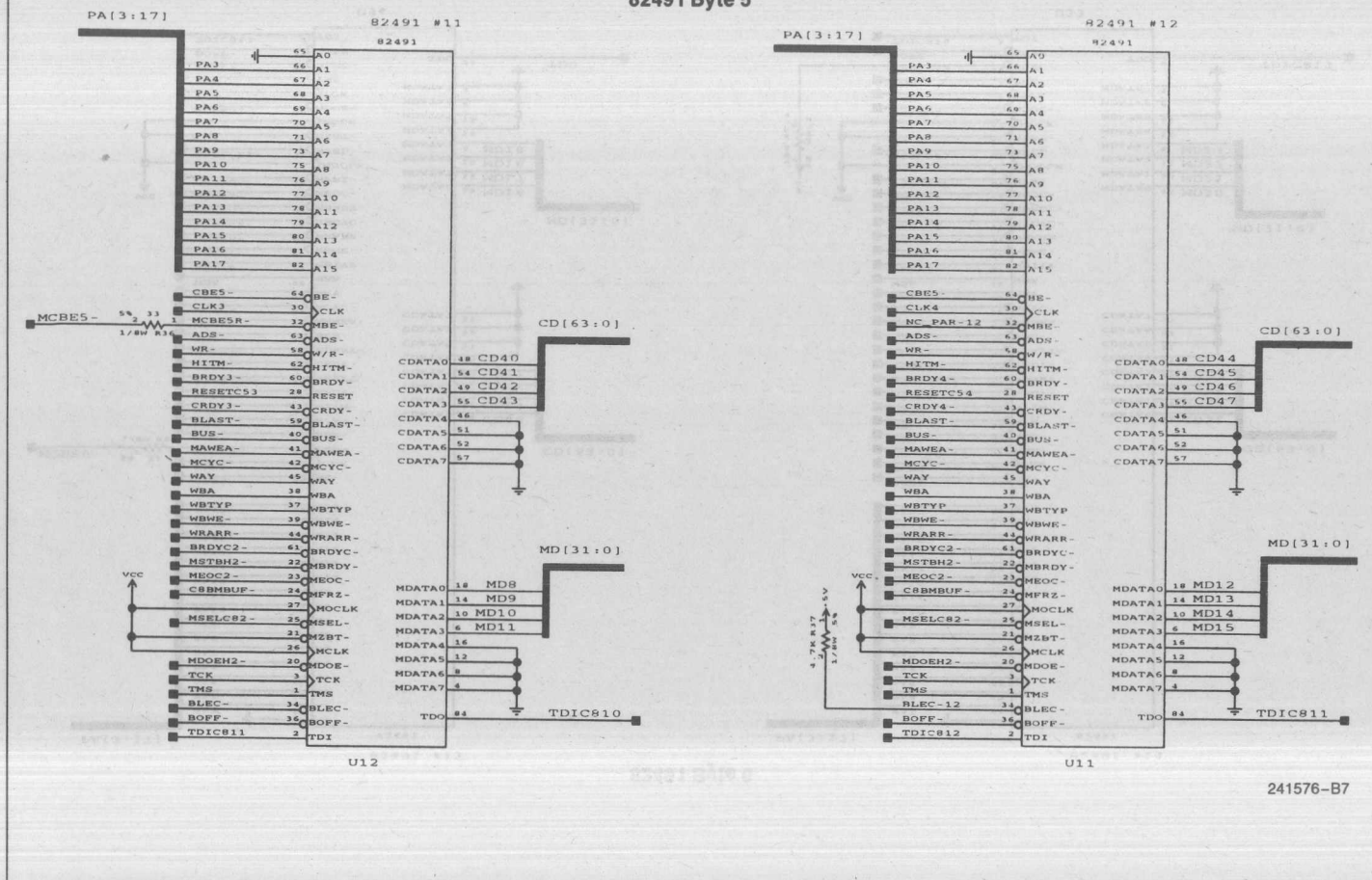


82491 Byte 4

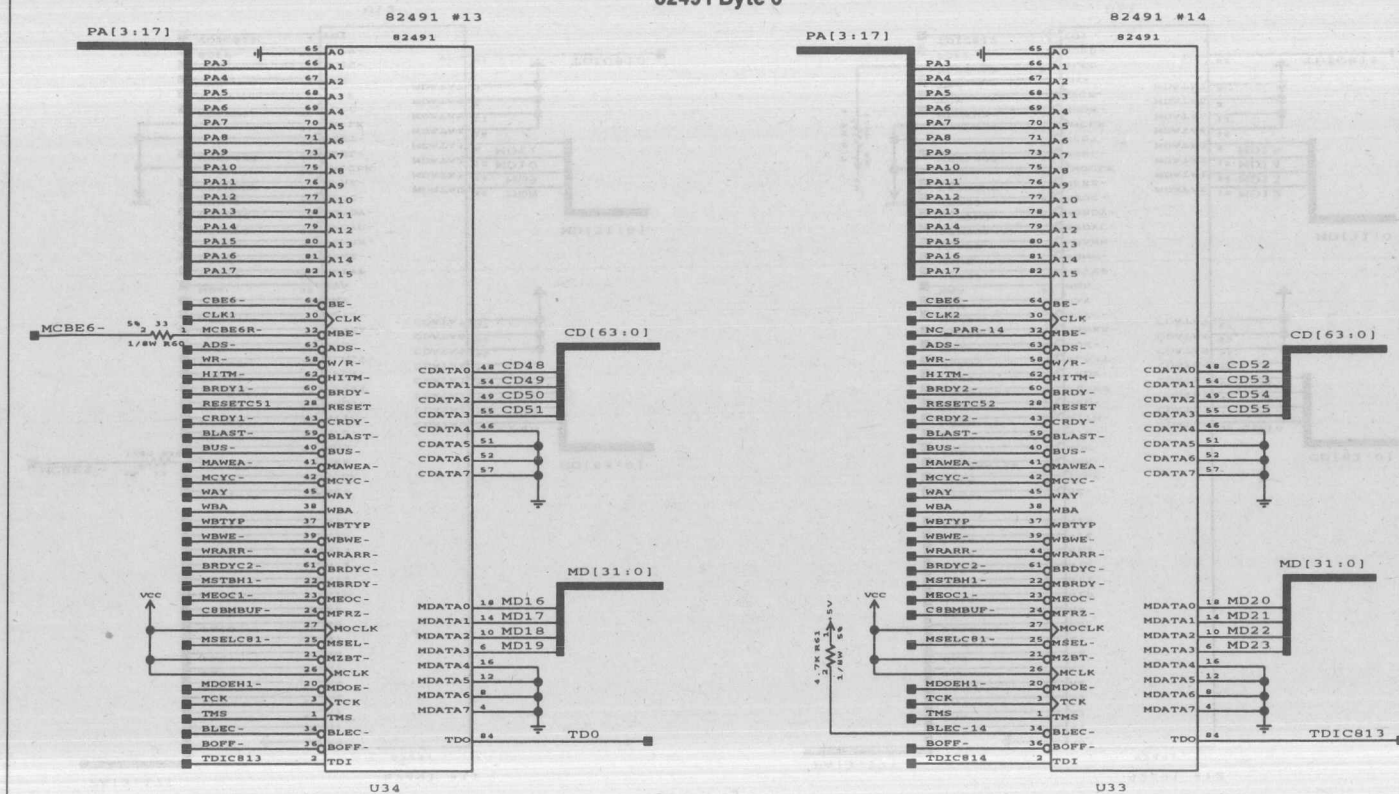


241576-B6

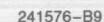
82491 Byte 5



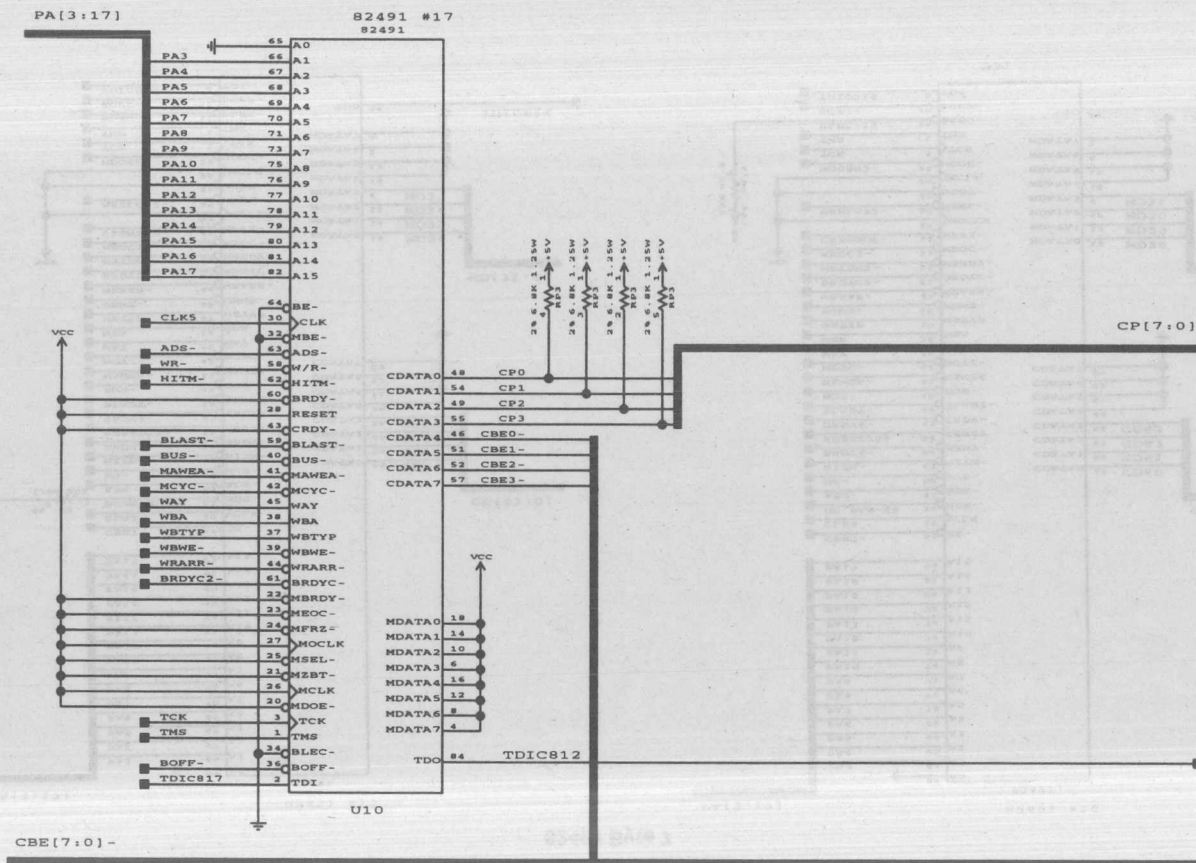
82491 Byte 6



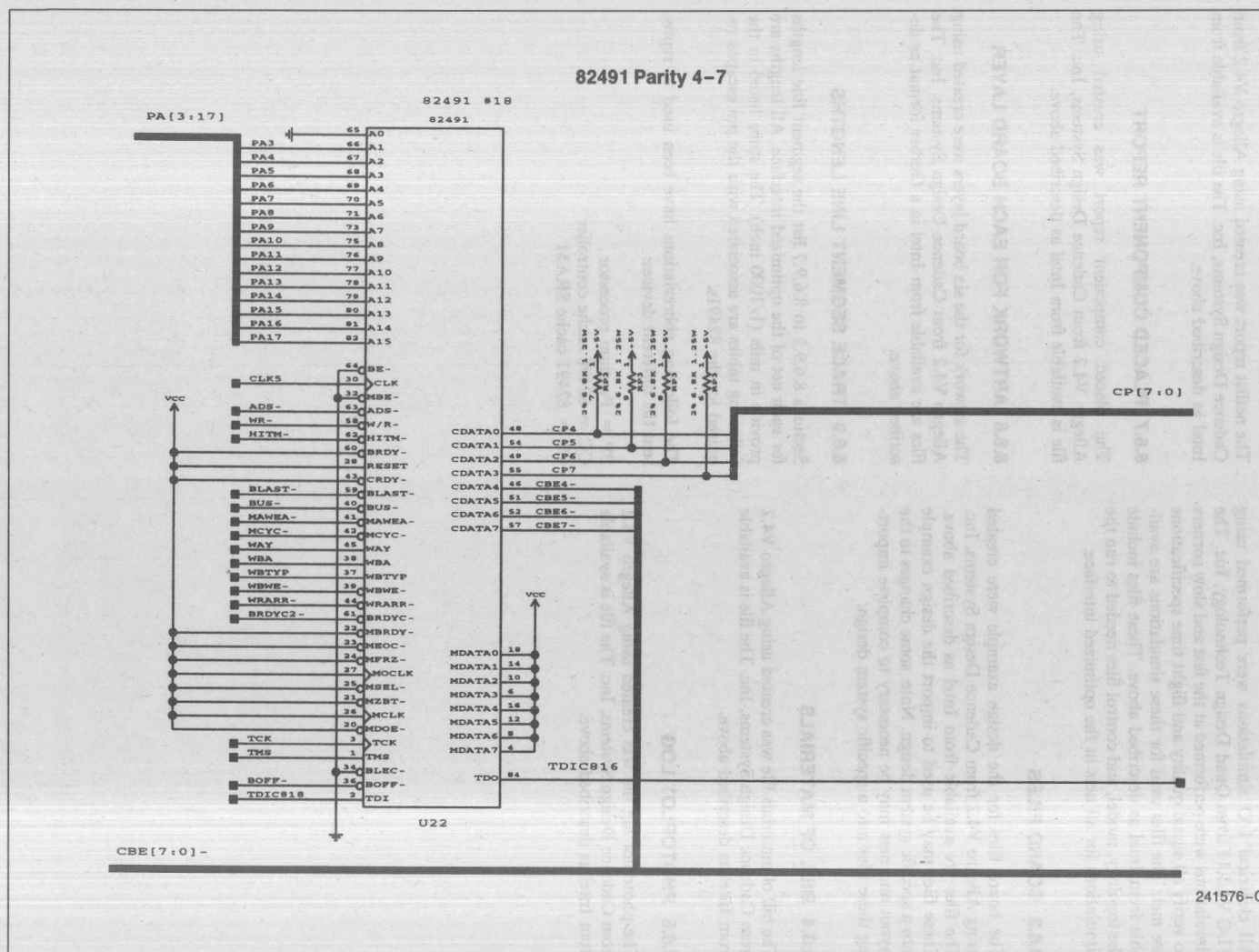
3



82491 Parity 0-3



241576-C0



8.6.2 I/O MODEL FILES

All electrical I/O simulations were performed using TLC V4.1.13 from Quad Design Technology, Inc. The simulations were performed at the fast and slow corners to verify all signal quality and flight time specifications are met. The files used for these simulations are available from Intel as described above. These files include the topology, model, and control files needed to run the simulations for all nets in the optimized interface.

8.6.3 BOARD FILES

The board files for the design example were created using Allegro V4.2 from Cadence Design Systems, Inc. The files are available from Intel as described above. These files may be used to import the design example into a specific system design. Note: some changes to the layout and nets may be necessary to complete importing these files into a specific system design.

8.6.4 BILL OF MATERIALS

The bill of materials file was created using Allegro V4.2 from Cadence Design Systems, Inc. The file is available from Intel as described above.

8.6.5 PHOTOPLOT LOG

The photoplot log file was created using Allegro V4.2 from Cadence Design Systems, Inc. The file is available from Intel as described above.

8.6.6 NETLIST REPORT

The netlist report was created using Allegro V4.2 from Cadence Design Systems, Inc. The file is available from Intel as described above.

8.6.7 PLACED COMPONENT REPORT

The placed component report was created using Allegro V4.2 from Cadence Design Systems, Inc. The file is available from Intel as described above.

8.6.8 ARTWORK FOR EACH BOARD LAYER

The artwork for the six board layers were created using Allegro V4.2 from Cadence Design Systems, Inc. The files are available from Intel in a Gerber format as described above.

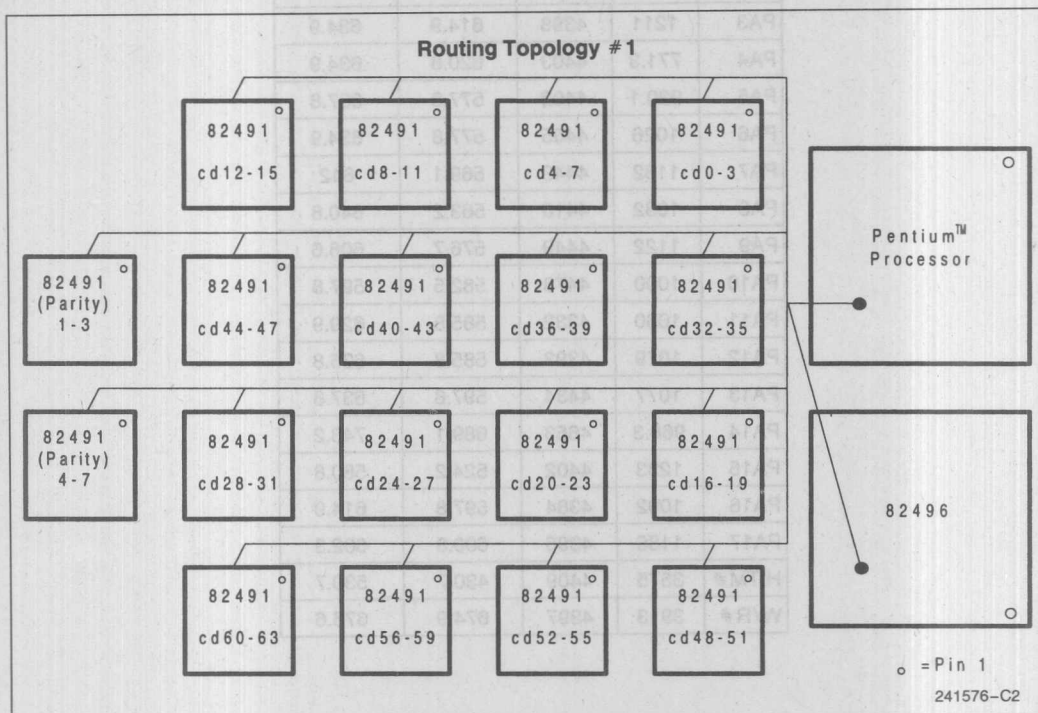
8.6.9 TRACE SEGMENT LINE LENGTHS

Sections 8.6.9.1 to 8.6.9.7 list the segment line lengths for each net of the optimized interface. All lengths are provide in mils (1/1000 inch). The stubs listed in the following tables are associated with the pin escapes required for the 82491s.

The following abbreviations have been used to represent the different devices:

PP = Pentium processor
CC = 82496 cache controller
CS = 82491 cache SRAM

8.6.9.1 Low Addresses and Pentium™ Processor Control



3

NET	PP-Tee	CC-Tee	Tee-CS#5	Tee-CS#9
PA3	1211	4398	614.9	634.9
PA4	771.3	4403	620.8	634.9
PA5	920.1	4402	577.8	637.8
PA6	1026	4405	577.8	634.9
PA7	1132	4443	569.1	642
PA8	1032	4410	563.2	640.8
PA9	1122	4449	576.7	606.6
PA10	1000	4478	582.5	597.8
PA11	1060	4332	585.8	629.9
PA12	1019	4393	585.8	635.8
PA13	1077	4434	597.8	637.8
PA14	968.3	4653	689.1	743.2
PA15	1233	4402	524.2	580.8
PA16	1092	4384	597.8	614.9
PA17	1186	4396	600.8	602.3
HITM#	3575	4409	430.7	530.7
W/R#	3913	4397	674.9	676.6

NET	CS # 9- CS # 1	CS # 1- CS # 2	CS # 2- CS # 3	CS # 3- CS # 4	CS # 9- CS # 10	CS # 10- CS # 11	CS # 11- CS # 12	CS # 12- CS # 17
PA3	1111	944.9	936.6	961.4	1124	936.6	944.9	936.6
PA4	1210	964.9	936.6	975.6	1207	936.6	961.9	936.6
PA5	1131	944.9	936.6	961.4	1104	936.6	944.9	936.6
PA6	1165	936.6	936.6	936.6	1177	936.6	936.6	936.6
PA7	1170	936.6	936.6	936.6	1157	936.6	936.6	936.6
PA8	1170	936.6	936.6	936.6	1157	936.6	936.6	936.6
PA9	1224	936.6	936.6	936.6	1217	936.6	936.6	936.6
PA10	1211	936.6	936.6	936.6	1207	936.6	936.6	936.6
PA11	1264	936.6	936.6	936.6	1267	936.6	936.6	936.6
PA12	1264	936.6	936.6	936.6	1267	936.6	936.6	936.6
PA13	1293	936.6	936.6	936.6	1297	936.6	936.6	936.6
PA14	1295	936.6	936.6	936.6	1297	936.6	936.6	936.6
PA15	1312	936.6	936.6	936.6	1317	936.6	936.6	936.6
PA16	1309	936.6	936.6	936.6	1317	936.6	936.6	936.6
PA17	1297	936.6	936.6	936.6	1287	936.6	936.6	936.6
HITM #	1141	953.1	953.1	953.1	1147	936.6	996.9	944.9
W/R #	1193	953.1	953.1	1023	1192	936.6	1003	953.1

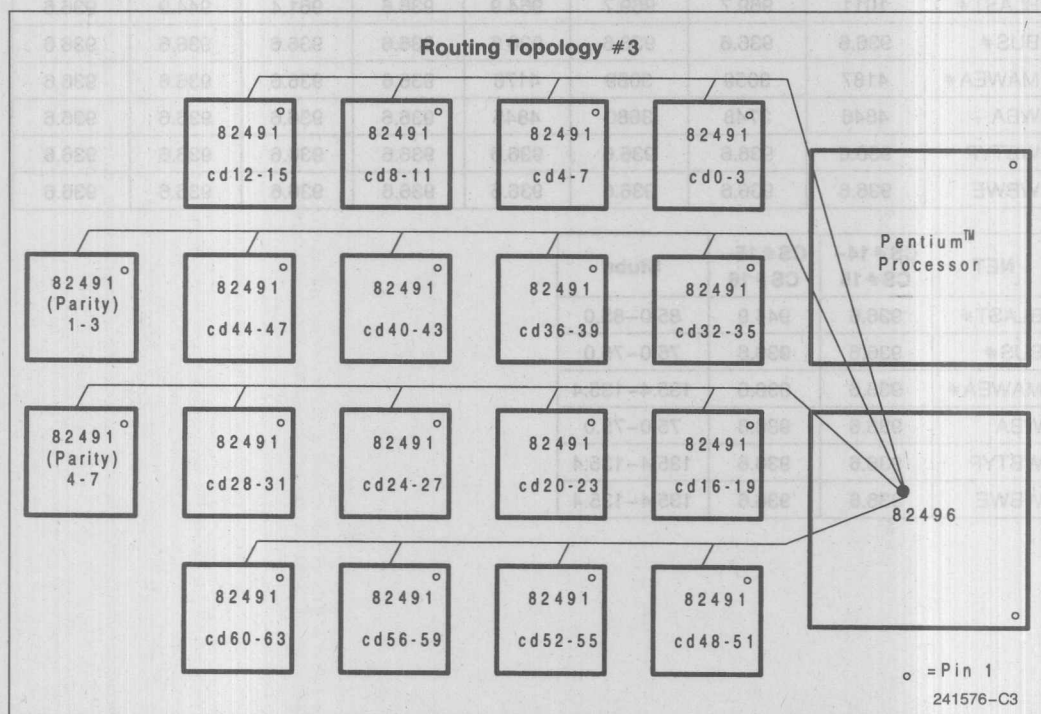
NET	CS #5- CS #6	CS #6-7 CS #	CS #7- CS #8	CS #8- CS #18	CS #5- CS #13	CS #13- CS #14	CS #14- CS #15	CS #15- CS #16	Stubs
PA3	1129	961.4	953.1	936.6	1122	944.9	936.6	944.9	75.0- 75.0
PA4	1207	969.7	967.3	944.9	1185	959	936.6	964.9	135.4- 135.4
PA5	1136	961.4	953.1	936.6	1145	944.9	936.6	944.9	75.0- 75.0
PA6	1190	936.6	936.6	936.6	1179	936.6	936.6	936.6	135.4- 135.4
PA7	1176	936.6	936.6	936.6	1179	936.6	936.6	936.6	75.0- 75.0
PA8	1177	936.6	936.6	936.6	1176	936.6	936.6	936.6	135.4- 135.4
PA9	1217	936.6	936.6	936.6	1213	936.6	936.6	936.6	135.4- 135.4
PA10	1207	936.6	936.6	936.6	1208	936.6	936.6	936.6	135.4 -135.4
PA11	1267	936.6	936.6	936.6	1261	936.6	936.6	936.6	75.0- 75.0
PA12	1267	936.6	936.6	936.6	1261	936.6	936.6	936.6	135.4 -135.4
PA13	1297	936.6	936.6	936.6	1281	936.6	936.6	936.6	75.0- 75.0
PA14	1297	936.6	936.6	936.6	1284	936.6	936.6	936.6	135.4 -135.4
PA15	1317	936.6	936.6	936.6	1318	936.6	936.6	936.6	75.0- 75.0
PA16	1317	936.6	936.6	936.6	1315	936.6	936.6	936.6	135.4 -135.4
PA17	1287	936.6	936.6	936.6	1288	936.6	936.6	936.6	75.0-75.0
HITM#	1146	944.9	944.9	964.9	1138	944.9	936.6	936.6	95.4 -95.4
W/R#	1197	975.6	961.4	953.1	1193	953.1	936.6	936.6	95.4 -95.4

NET	CC-Tee	PP-Tee	Tee-CS#5	Tee-CS#9	CS#9-CS#1	CS#1-CS#2	CS#2-CS#3	CS#3-CS#4
BOFF#	2405.6	4401.9	919.7	925.8	936.6	936.6	936.6	936.6

NET	CS#9-CS#10	CS#10-CS#11	CS#11-CS#12	CS#12-CS#17	CS#5-CS#6	CS#6-CS#	CS#7-CS#8	CS#8-CS#18
BOFF#	936.6	936.6	936.6	936.6	936.6	936.6	936.6	936.6

NET	CS#5-CS#13	CS#13-CS#14	CS#14-CS#15	CS#15-CS#16	Stubs
BOFF#	944.9	936.6	936.6	936.6	75.0-75.0

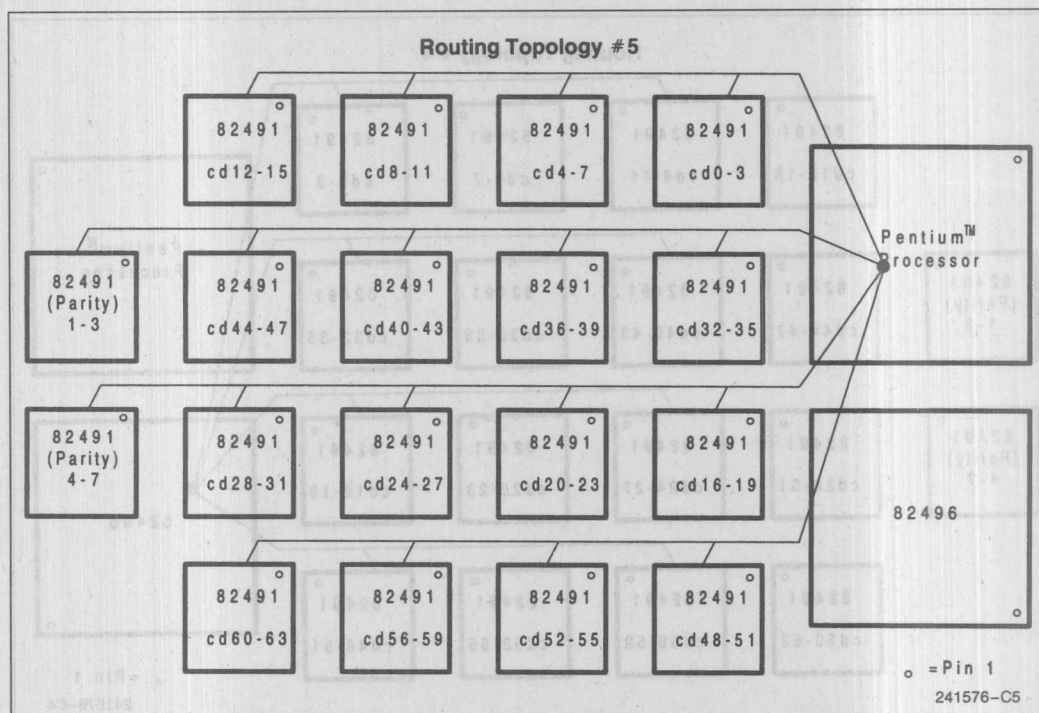
8.6.9.2 82496 Control



NET	CC- CS #1	CC- CS #9	CC- CS #5	CC- CS #13	CS #1- CS #2	CS #2- CS #3	CS #3- CS #4	CS #9- CS #10
BLAST #	4225	3085	3089	4186	961.4	967.3	961.4	1020
BUS #	4318	3216	3210	4311	936.6	936.6	936.6	936.6
MAWEA #	936.6	936.6	936.6	936.6	936.6	936.6	936.6	936.6
WBA	936.6	936.6	936.6	936.6	936.6	936.6	936.6	936.6
WBTYP	4087	2986	2974	4086	936.6	936.6	936.6	936.6
WBWE	4210	3109	3113	4215	936.6	936.6	936.6	936.6

NET	CS #10- CS #11	CS #11- CS #12	CS #12- CS #17	CS #5- CS #6	CS #6- CS #7	CS #7- CS #8	CS #8- CS #18	CS #13- CS #14
BLAST #	1011	969.7	969.7	964.9	936.6	961.4	944.9	936.6
BUS #	936.6	936.6	936.6	936.6	936.6	936.6	936.6	936.6
MAWEA #	4187	3059	3089	4178	936.6	936.6	936.6	936.6
WBA	4846	3748	3680	4845	936.6	936.6	936.6	936.6
WBTYP	936.6	936.6	936.6	936.6	936.6	936.6	936.6	936.6
WBWE	936.6	936.6	936.6	936.6	936.6	936.6	936.6	936.6

NET	CS #14- CS #15	CS #15- CS #16	Stubs
BLAST #	936.6	944.9	85.0-85.0
BUS #	936.6	936.6	75.0-75.0
MAWEA #	936.6	936.6	135.4-135.4
WBA	936.6	936.6	75.0-75.0
WBTYP	936.6	936.6	135.4-135.4
WBWE	936.6	936.6	135.4-135.4

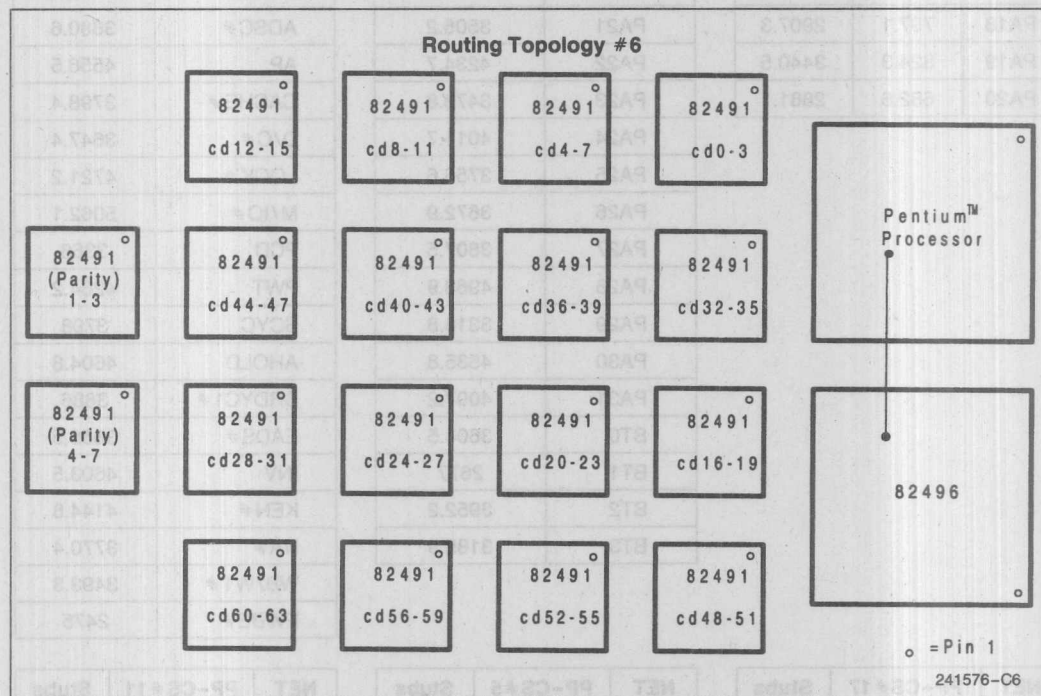


NET	PP- CS#1	PP- CS#9	PP- CS#5	PP- CS#13	CS#1- CS#2	CS#2- CS#3	CS#3- CS#4	CS#9- CS#10	CS#10- CS#11
ADS#	5213.8	4111.6	4108.4	5207.4	936.6	936.6	936.6	964.9	1003.8

NET	CS#11- CS#12	CS#12- CS#17	CS#5- CS#6	CS#6- CS#7	CS#7- CS#8	CS#8- CS#18	CS#13- CS#14	CS#14- CS#15	CS#15- CS#16
ADS#	964.9	978	936.6	936.6	936.6	969	961.9	959	961.9

NET	Stubs
ADS#	75.0-75.0

8.6.9.5 Pentium™ Processor and 82496 Control, High Addresses, Pentium™ Processor Data



3

NET	PP-res	res-CC
PA18	797.1	2907.3
PA19	824.3	3440.5
PA20	682.6	2881.1

NET	PP-CC
PA21	3505.2
PA22	4234.7
PA23	3478.8
PA24	4011.7
PA25	3756.6
PA26	3672.9
PA27	3807.5
PA28	4963.9
PA29	3318.8
PA30	4535.8
PA31	4097.2
BT0	3604.5
BT1	2677
BT2	3952.2
BT3	3185.9

NET	PP-CC
ADSC #	3880.6
AP	4556.5
CACHE #	3798.4
D/C #	3647.4
LOCK #	4721.2
M/IO #	5062.1
PCD	3366
PWT	4264.2
SCYC	3798
AHOLD	4604.8
BRDYC1 #	3686
EADS #	3656.5
INV	4603.5
KEN #	4144.8
NA #	3770.4
WB/WT #	3493.3
EWBE #	2475

NET	PP-CS # 17	Stubs
CP0	7558	135.4
CP1	7685.9	135.4
CP2	7675.3	75
CP3	7307.4	75

NET	PP-CS # 5	Stubs
CD16	7121.5	135.4
CD17	6878.5	135.4
CD18	6974.5	75
CD19	7021.5	75

NET	PP-CS # 11	Stubs
CD40	692.8	135.4
CD41	7567.4	135.4
CD42	6922.3	75
CD43	7613.3	75

NET	PP-CS # 18	Stubs
CD4	7641.3	135.4
CD5	7698.3	135.4
CD6	7416.4	75
CD7	6946.9	75

NET	PP-CS # 6	Stubs
CD20	7089.3	135.4
CD21	6948.1	135.4
CD22	7064.9	75
CD23	6927.4	75

NET	PP-CS # 12	Stubs
CD44	7018	95.4
CD45	6997.1	135.4
CD46	6956.1	75
CD47	7491.6	75

NET	PP-CS #1	Stubs
CD0	6920.6	135.4
CD1	6964.1	135.4
CD2	7130.1	75
CD3	6910.9	75

NET	PP-CS #7	Stubs
CD24	6968.1	107.1
CD25	6877.9	135.4
CD26	7040.2	75
CD27	6891.7	75

NET	PP-CS #13	Stubs
CD48	6961	135.4
CD49	6884.2	135.4
CD50	7072.2	75
CD51	6968.3	75

NET	PP-CS #2	Stubs
CD4	7037.3	135.4
CD5	6869.4	135.4
CD6	6925	75
CD7	6886.5	75

NET	PP-CS #31	Stubs
CD28	7688.3	135.4
CD29	7872	135.4
CD30	7395.9	75
CD31	7433.3	75

NET	PP-CS #1	Stubs
CD52	7277.5	135.4
CD53	6979.9	135.4
CD54	6921.7	75
CD55	6920.2	75

NET	PP-CS #1	Stubs
CD8	6945.5	135.4
CD9	7448.5	135.4
CD10	7790.2	75
CD11	6924.1	75

NET	PP-CS #9	Stubs
CD32	7586.8	135.4
CD33	7271.6	135.4
CD34	7424.6	75
CD35	6944.1	75

NET	PP-CS #15	Stubs
CD56	6777.1	135.4
CD57	6997.1	135.4
CD58	6881.1	75
CD59	7130.3	75

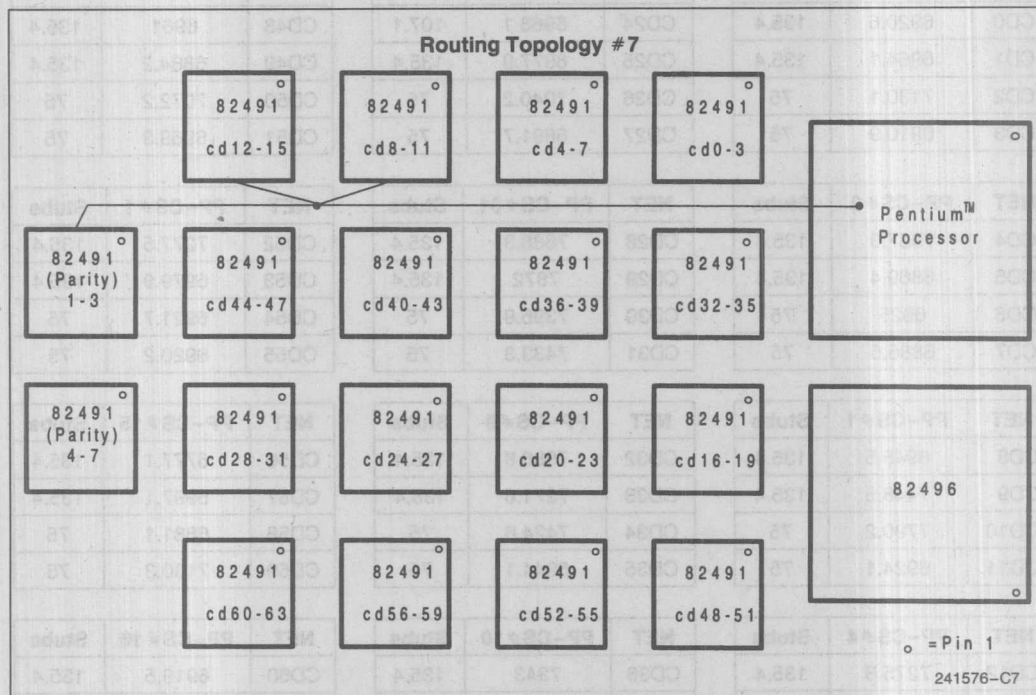
NET	PP-CS #4	Stubs
CD12	7275.6	135.4
CD13	7344.7	135.4
CD14	7692.1	75
CD15	7438.1	75

NET	PP-CS #10	Stubs
CD36	7343	135.4
CD37	6952.2	135.4
CD38	7520	75
CD39	7403	75

NET	PP-CS #16	Stubs
CD60	6919.5	135.4
CD61	6971.3	135.4
CD62	7274.5	75
CD63	7019.4	75

3

8.6.9.6 Byte Enables



NET	PP-Tee	Tee-CS # 1	Tee-CS # 2	Tee-CS # 17	Stubs
CBE0 #	1552.2	708.5	707.3	4813.3	75.0-135.4

NET	PP-Tee	Tee-CS # 3	Tee-CS # 4	Tee-CS # 17	Stubs
CBE1 #	4358.7	545.6	543.1	2627.4	75.0-75.0

NET	PP-Tee	Tee-CS # 5	Tee-CS # 6	Tee-CS # 17	Stubs
CBE2 #	3476.4	562.1	563.1	4330.5	75.0-135.4

NET	PP-Tee	Tee-CS# 7	Tee-CS# 8	Tee-CS# 17	Stubs
CBE3#	5352.9	537.3	440.7	3011.3	75.0-75.0

NET	PP-Tee	Tee-CS# 9	Tee-CS# 10	Tee-CS# 18	Stubs
CBE4#	2709.8	520.7	440.7	4762.9	75.0-135.4

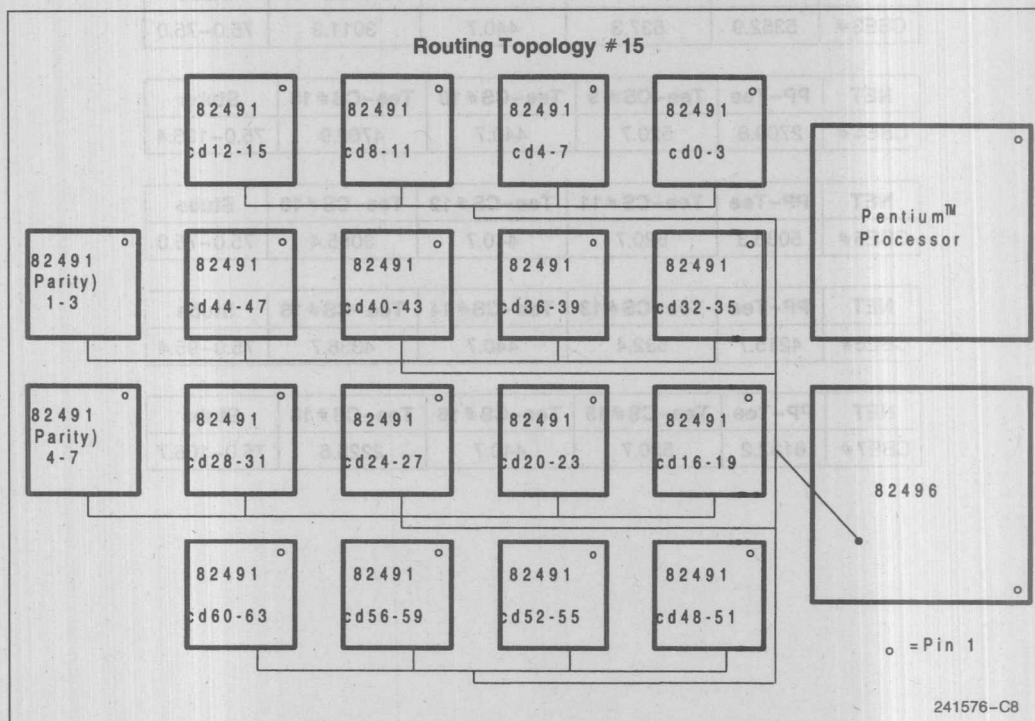
NET	PP-Tee	Tee-CS# 11	Tee-CS# 12	Tee-CS# 18	Stubs
CBE5#	5035.3	520.7	440.7	3065.4	75.0-75.0

NET	PP-Tee	Tee-CS# 13	Tee-CS# 14	Tee-CS# 18	Stubs
CBE6#	4215.7	532.4	440.7	4338.7	75.0-95.4

NET	PP-Tee	Tee-CS# 15	Tee-CS# 16	Tee-CS# 18	Stubs
CBE7#	6132.2	520.7	440.7	2225.6	75.0-105.7

3

8.6.9.7 82496 Control



NET	CC-Tee	Tee-CS#9	Tee to CS#5	CS#5 to CS#13	CS#1 to CS#9	CS#1-CS#2	CS#2-CS#3	CS#3-CS#4	CS#9-CS#10	CS#10-CS#11
BRDYC2	2077	598	595	935	932	940	940	940	540	940
MCYC	1707	540	527	1038	1041	940	940	940	1047	940
WRARR#	1820	573	576	976	976	940	940	940	940	940
WAY	1969	749	747	944	944	940	940	940	940	940

NET	CS # 11- CS # 12	CS # 12- CS # 17	CS # 5- CS # 6	CS # 6- CS # 7	CS # 7- CS # 8	CS # 8- CS # 18	CS # 13- CS # 14	CS # 14- CS # 15	CS # 15- CS # 16	Stubs
BRDYC2	940	940	940	940	940	940	940	940	940	85
MCYC	940	940	1047	940	940	940	940	940	940	75
WRARR #	940	940	940	940	940	940	940	940	940	135
WAY	940	940	940	940	940	940	940	940	940	75

References

1. Intel Corporation, *Pentium™ Processor User's Manual*, Order #: 241563.
2. Intel Corporation, *Pentium™ Processor Thermal Design Guide*, Application Note: AP-480, Order# 241575.
3. Intel Corporation, *Pentium Processor Clock Design*, Application Note: AP-479, Order# 241574.
4. Blood, William R., Jr., *MECL System Design Handbook*, 1988, Motorola Inc.
5. T. Rahal-Arabi and R. Suarez-Gartner, "An Efficient Methodology for the Design of Optimum Electrical Performance of Interconnects in High Performance Systems," IEEE Topical Meeting on Electrical Performance of Electronic Packaging, Tucson AZ, April 1992.
6. Huck, Scott, "Simulating Intel's Optimized Interface with the Intel486™ DX CPU-Cache Chip-Set," 1992, Intel Corp.

[illegible]

Intel Processor Identification with the CPUID Instruction

PRELIMINARY

Order Number: 241618-001

Intel Processor Identification with the CPUID Instruction

CONTENTS	PAGE
1.0 BACKGROUND	3-344
2.0 DETECTING THE CPUID INSTRUCTION	3-344
3.0 OUTPUTS OF THE CPUID INSTRUCTION	3-344
3.1 Vendor-ID String	3-344
3.2 CPU Signature	3-344
3.3 Feature Flags	3-347
4.0 USAGE GUIDELINES	3-347
5.0 PROPER IDENTIFICATION SEQUENCE	3-347

CONTENTS	PAGE
EXAMPLES	
Example 1. CPUID Identification Procedure	3-349
FIGURES	
Figure 1. CPUID Instruction Outputs	3-345
Figure 2. CPU Signature Format on Intel386™ Processors	3-346
Figure 3. Flow of GET_CPUID Procedure	3-348
TABLES	
Table 1. Effects of EAX Contents on CPUID	3-345
Table 2. Intel486™ and Pentium™ Processor Signatures	3-346
Table 3. Intel386™ CPU Signatures	3-346
Table 4. Feature Flag Values	3-347

3

Beginning with the Intel386 processor, the CPUID instruction has been available at user level. With processors that implement the CPUID instruction, the CPU signature is made available both by the CPUID instruction (Figure 1) and by the CPUID instruction (Figure 2). The Intel486 processor and later processors implement the CPUID instruction. Table 1 shows the values that are currently defined. (The high-order 20 bits are undefined and reserved.)

On Intel386 processors, the format of the CPU signature is somewhat different. Figure 2 shows Table 2, which gives the currently possible values.

Regardless of signature format, the MODEL and FAMILY fields are significant only in processors that do not implement the CPUID instruction.

The STEPPING field helps software deal with errors. Intel releases information about errors and related stepping numbers as needed.

2.0 DETECTING THE CPUID INSTRUCTION

Intel has provided a straightforward method for detecting whether the CPUID instruction is available. This method uses the ID flag in bit 21 of the EFLAGS register. If the ID flag is set, the value of the CPUID instruction is available. The program example in Section 2.0 shows how to use the PUSHFD instruction to read and test the ID flag.

3.0 OUTPUTS OF THE CPUID INSTRUCTION

Figure 1 summarizes the output of CPUID. The CPUID instruction can be executed multiple times with a different parameter in the EAX register. The outputs depend on the value of EAX as follows:

1.0 BACKGROUND

As new generations and new models of processors have been added to the Intel X86 architecture (8086, 8088, Intel 286, Intel386™, Intel486™, and Pentium™ processors), Intel has provided increasingly sophisticated methods to software for identifying the features available on the processor.

- First, Intel started publishing code sequences that identify processor generation by detecting minor differences of implementation.
- Later, with the advent of the Intel386 processor, Intel started providing the CPU signature (family, model, and stepping numbers) to software at reset.
- Ultimately, Intel has extended the X86 architecture with the CPUID instruction. The CPUID instruction not only provides the CPU signature but also provides information about the features supported by the processor.

This latest step is necessary because the computing market is demanding processor models within a given processor generation that have differing sets of features. Anticipating that this trend will continue with future processor generations, Intel has made the CPUID instruction extensible.

The purpose of this Application Note is to show how to use the CPUID instruction in such a way that software can execute compatibly on the widest possible range of Intel X86 generations and models, past, present, and future.

2.0 DETECTING THE CPUID INSTRUCTION

Intel has provided a straightforward method for detecting whether the CPUID instruction is available. This method uses the ID flag in bit 21 of the EFLAGS register. If software can change the value of this flag, the CPUID instruction is available. The program example in Section 5.0 shows how to use the PUSHFD instruction to read and the POPFD instruction to change the value of the ID flag.

3.0 OUTPUTS OF THE CPUID INSTRUCTION

Figure 1 summarizes the outputs of CPUID.

The CPUID instruction can be executed multiple times, each time with a different parameter in the EAX register. The outputs depend on the value of EAX, as

Table 1 specifies. To determine the highest acceptable value of the EAX parameter, the program should set EAX to zero. In this case, CPUID returns to EAX the value of the highest parameter that it recognizes. No execution of CPUID should use a parameter greater than this highest value. Although this highest value is 1 for the first model of the Pentium processor, it might be different for future processor models.

3.1 Vendor-ID String

The vendor identification string is stored in the registers EBX, EDI, and ECX in such a way that, when the contents of these registers are stored in memory in adjacent locations (EBX at the lowest address, ECX at the highest), the byte string "GenuineIntel" appears in memory.

While any imitator of the X86 architecture can provide the CPUID instruction, no imitator can legitimately claim that its part is a genuine Intel part. Therefore, the presence of the "GenuineIntel" string is an assurance that the CPUID instruction and the CPU signature are implemented as described in this document.

3.2 CPU Signature

Beginning with the Intel386 processor, the CPU signature has been available at reset. With processors that implement the CPUID instruction, the CPU signature is made available both by reset and by the CPUID instruction. Figure 1 shows the format of the signature on the Intel486 processor, Pentium processor, and later processor generations; Table 2 shows the values that are currently defined. (The high-order 20 bits are undefined and reserved.)

On Intel386 processors, the format of the CPU signature is somewhat different, as Figure 2 shows. Table 3 gives the currently possible values.

Regardless of signature format, the MODEL and FAMILY fields are significant only in processors that do not implement the CPUID instruction.

The STEPPING fields help software deal with errata. Intel releases information about errata and related stepping numbers as needed.

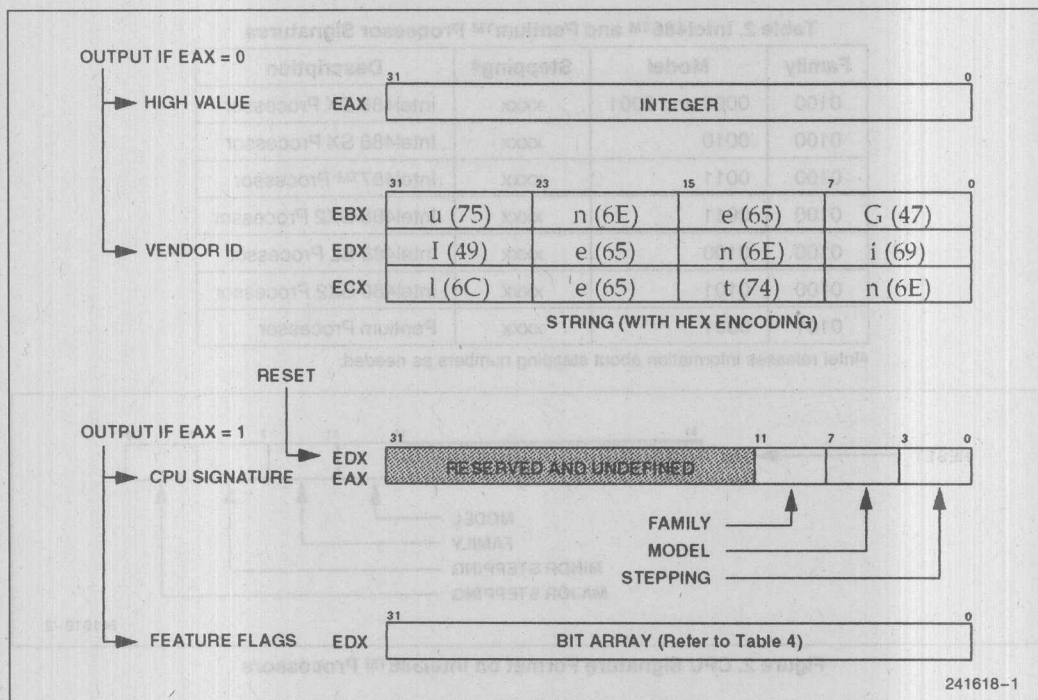


Figure 1. CPUID Instruction Outputs

Table 1. Effects of EAX Contents on CPUID

Parameter	Outputs of CPUID
EAX = 0	EAX ← highest value
	EBX:EDX:ECX ← Vendor Identification String
EAX = 1	EAX ← CPU signature
	EDX ← Feature flags
1 < EAX ≤ highest value	Might be defined in future processor models
EAX > highest value	Undefined

Family	Model	Stepping ^a	Description
0100	0000 and 0001	xxxx	Intel486 DX Processor
0100	0010	xxxx	Intel486 SX Processor
0100	0011	xxxx	Intel487™ Processor
0100	0011	xxxx	Intel486 DX2 Processor
0100	0100	xxxx	Intel486 SL Processor
0100	0101	xxxx	Intel486 SX2 Processor
0101	0001	xxxx	Pentium Processor

^aIntel releases information about stepping numbers as needed.

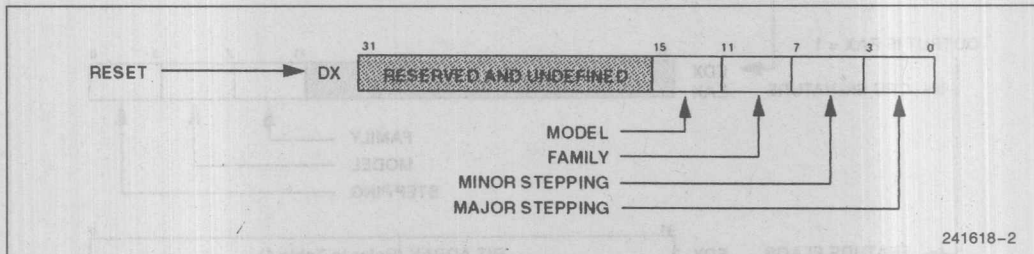


Figure 2. CPU Signature Format on Intel386™ Processors

Table 3. Intel386™ CPU Signatures

Model	Family	Major Stepping	Minor Stepping ^a	Description
0000	0011	0000	xxxx	Intel386 DX Processor
0010	0011	0000	xxxx	Intel386 SX Processor
0011	0011	0000	xxxx	Intel 376 Processor
0100	0011	0000	xxxx	Intel386 SL Processor
0100	0011	0001	xxxx	Intel386 SL Processor (B-step)
0000	0011	0100	xxxx	RapidCAD™ Processor

^aIntel releases information about minor stepping numbers as needed.

3.3 Feature Flags

The CPUID instruction sets the feature register, EDX, to indicate which features the processor supports. Each set feature flag can indicate either that a feature is present or that a feature is not present. Refer to Table 4 for the meanings of the feature flags that are currently defined. For each future processor generation or model, refer to the programmer's reference manual, user's manual, or equivalent documentation for additional definitions.

Software executing on a processor that supports the CPUID instruction should use the feature flags instead of the model and family to determine what features are present. Doing so helps make software immune to incompatibilities that might arise from the release of an additional model that has a feature set different than that of the model for which the software was designed.

Table 4. Feature Flag Values

Bit Position	Abbreviation	Meaning When Flag = 1
0	FPU	Floating-point unit on-chip
1-6 ^a	(see note)	(see note)
7	MCE	Machine-check exception present
8	CX8	CMPXCHG8B instruction present
9-31 ^a	(see note)	(see note)

^aSome non-essential information regarding the Pentium processor is considered Intel confidential and proprietary and has not been documented in this publication. This information is provided in the *Supplement to the Pentium™ Processor User's Manual* and is available with the appropriate non-disclosure agreements in place. Contact Intel Corporation for details.

4.0 USAGE GUIDELINES

This document presents straightforward -feature-detection methods. Software should not try to identify features by exploiting programming tricks, "undocumented features," or otherwise deviating from the intent of this document. The following list gives some tips that can help programmers maintain the widest range of compatibility for their software.

- Do not depend on the absence of an invalid opcode trap on the CPUID opcode to detect CPUID. Do not depend on the absence of an invalid opcode trap on the PUSHFD opcode to detect a 32-bit processor. Test the ID flag, as described in Section 2.0 and shown in Section 5.0.
- Do not assume that a given family or model has any specific feature. For example, do not assume that, because FAMILY = 5 (-Pentium processor), there must be a floating-point unit on-chip. Use the feature flags for this determination.

- Do not assume that the existence of the CPUID instruction implies a Pentium processor or later generation CPU. Future versions of the Intel486 microprocessor may also include the CPUID instruction.
- Do not use undocumented features of a CPU to identify steppings or features. For example, the Intel386 CPU A-step had bit instructions that were withdrawn with B-step. Some software attempted to execute these instructions and depended on the invalid-opcode exception as a signal that it was not running on the A-step part. This software failed to work correctly when the Intel486 CPU used the same opcodes for different instructions. That software should have used the stepping information in the CPU signature.
- Do not assume that a value of 1 in a feature flag indicates that a given feature is present, even though that is the case in the first model of the Pentium processor. For some feature flags that might be defined in the future, a value of 1 can indicate that the corresponding feature is not present.
- Programmers should be careful to test feature flags individually and not make assumptions about irrelevant bits. It would be a mistake, for example, to test the FPU bit by comparing the feature register to 1 with a compare instruction.
- Do not assume that the clock of a given family or model runs at a specific speed. In particular, do not write clock-dependent code, such as timing loops. An upgrade processor can run at a faster speed, while reporting the same family and model. For an example, refer to the Intel487 and Intel486DX processors in Table 2. These processors have different clock speeds, even though they have the same family and model numbers. Use the systems timers for measuring time.

5.0 PROPER IDENTIFICATION SEQUENCE

The following program example concludes this document by showing correct usage of the CPUID instruction. It also shows how to identify earlier processor generations that implement neither the CPU signature nor the CPUID instruction. This program contains three procedures:

1. **get_cpuid** which identifies the type of CPU. Figure 3 shows the flow of this procedure.
2. **check_fpu** which determines what type of floating-point unit (FPU) or math coprocessor (MCP) is present.
3. **print** which uses DOS function calls to display the results of the other procedures on the monitor of a PC. This procedure can be omitted or modified to execute with other operating systems.

This procedure has been tested with 8086, 80286, Intel386, Intel486, and Pentium processors.

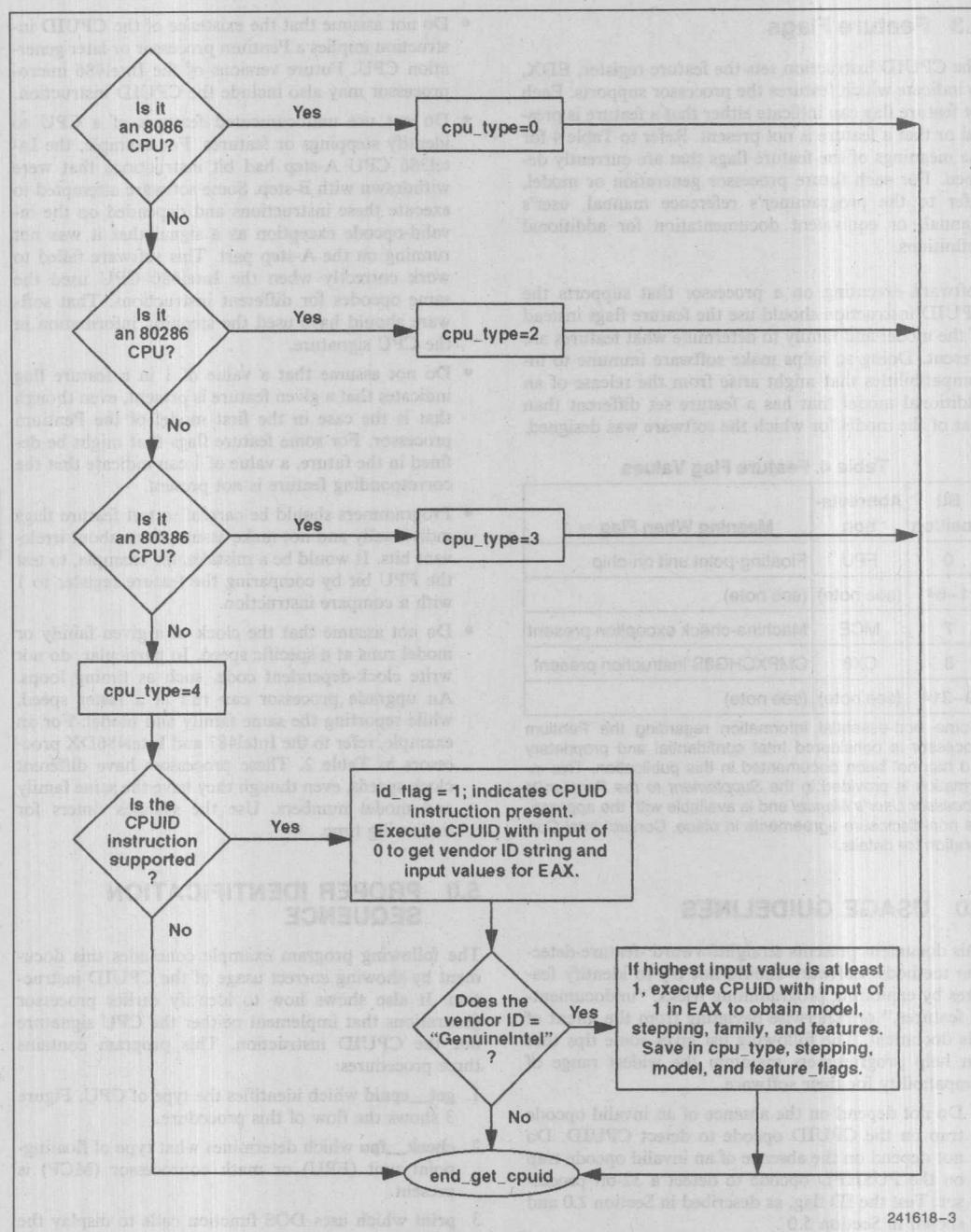


Figure 3. Flow of GET.CPUID Procedure

```
;
; Filename:      cpuid32.msm
;
; This program has been developed by Intel Corporation. You have
; Intel's permission to incorporate this source code into your
; product royalty free.
;
; Intel specifically disclaims all warranties, express or implied,
; and all liability, including consequential and other indirect
; damages, for the use of this code, including liability for
; infringement of any proprietary rights. Intel does not assume
; any responsibility for any errors which may appear in this code
; nor any responsibility to update it.
;
; This program contains three parts:
; Part 1: Identifies CPU type in the variable cpu_type:
;         0=8086 processor
;         2=Intel 286 processor
;         3=Intel386(TM) processor
;         4=Intel486(TM) processor
;         5=Pentium(TM) processor
;
; Part 2: Identifies FPU type in the variable fpu_type:
;         0=FPU not present
;         1=FPU present
;         2=287 present (only if cpu_type=3)
;         3=387 present (only if cpu_type=3)
;
; Part 3: Prints out the appropriate message. This part can
; be removed if this program is not used in a DOS-based
; system. Portions affected are at the end of the
; data segment and the print procedure in the code
; segment.
;
; This program was assembled with Microsoft's Assembler MASM 6.0.
; While this program mostly uses 16-bit operands, some 32-bit
; operands are required to check the 32-bit EFLAGS register once
; it has been determined that the processor is a least an
; Intel386 processor. 32-bit operations are invoked by using
; the macro OPND32.
;
; TITLE  CPUID
; DOSSEG
; .model  small
; .stack  100h
; .186
;
; The OPND32 macro takes either zero or two parameters.
; With zero parameters, it generates the 32-bit operand-size prefix.
; With two parameters, it generates the 32-bit operand-size prefix,
; followed by an opcode and a 32-bit immediate value. These parameters
; are used to generate XOR AX,imm32 instructions.
```

Example 1. CPU Identification Procedure


```

OPND32 MACRO op_code, op_eraud
    db      66h      ; Force 32-bit operand size
    IFNB <op_code>
        db      op_code ; Optional opcode
    IFNB <op_eraud>
        dd      op_eraud ; Optional 32-bit immediate value
    ENDIF
ENDIF
ENDM

CPUID MACRO
    db      0fh      ; Opcode for CPUID instruction
    db      0a2h
ENDM

TRUE equ 1
FAMILY_MASK equ 0f00h
FAMILY_SHIFT equ 8
MODEL_MASK equ 0f0h
MODEL_SHIFT equ 4
STEPPING_MASK equ 0fh
FPU_FLAG equ 1h
MCE_FLAG equ 80h
CMPXCHG8B_FLAG equ 100h

.data
fp_status dw ?
vendor_id db 12 dup (?)
cpu_type db ?
model db ?
stepping db ?
id_flag db 0
fpu_type db 0
intel_proc db 0
feature_flags dw 2 dup (0)
;
; Remove the remaining data declarations if not using the DOS-based
; print procedure
;

id_msg db "This system has a$"
fp_8087 db " and an 8087 math coprocessor$"
fp_80287 db " and an 80287 math coprocessor$"
fp_80387 db " and an 80387 math coprocessor$"
c8086 db "n 8086/8088 processor$"
c286 db "n 80286 processor$"
c386 db "n 80386 processor$"
c486 db "n 80486 DX processor or 80487 SX math coprocessor$"
c486nfp db "n 80486 SX processor$"
Intel486_msg db 13,10,"This system contains a Genuine "
db "Intel486(TM) processor",13,10,"$"
Pentium_msg db 13,10,"This system contains a Genuine "
db "Intel Pentium(TM) processor",13,10,"$"
modelmsg db "Model: $"
steppingmsg db "Stepping: $"

```

Example 1. CPU Identification Procedure (Continued)

```

familymsg      db 13,10,"Processor Family: $"
period         db " ",13,10,"$"
dataCR         db ?,13,10,"$"
intel_id       db "GenuineIntel"
fpu_msg        db 13,10,"This processor contains a FPU",13,10,"$"
mce_msg        db "This processor supports the "
               db "Machine Check Exception",13,10,"$"
cmp_msg        db "This processor supports the "
               db "CMPXCHG8B instruction",13,10,"$"
not_intel      db "It least an 80486 processor.",13,10
               db "It does not contain a Genuine Intel part and as a "
               db "result,",13,10,"the CPUID detection information "
               db "cannot be determined at this time.",13,10,"$"

```

```

;
;   This code identifies the processor and coprocessor
;   that are currently in the system. The program first
;   determines the processor id. When that is accomplished,
;   the program then determines whether a coprocessor
;   exists in the system. If a coprocessor or integrated
;   coprocessor exists, the program identifies
;   the coprocessor id. The program then prints out
;   the CPU and floating point presence and type.
;

```

```

.code
start: mov     ax, @data
      mov     ds, ax      ; set segment register
      mov     es, ax      ; set segment register
      pushf                    ; save for restoration at end
      call    get_cpuid
      call    get_fpuid
      call    print
      popf
      mov     ax, 4c00h      ; terminate program
      int     21h

```

```

get_cpuid proc
;
;   This procedure determines the type of CPU in a system
;   and sets the cpu_type variable with the appropriate value.
;   All registers are used by this procedure, none are preserved.
;
;   Intel 8086 CPU check
;   Bits 12-15 of the FLAGS register are always set on the
;   8086 processor.
;

```

Example 1. CPU Identification Procedure (Continued)

```

check_8086:
    pushf                ; push original FLAGS
    pop                 ax      ; get original FLAGS
    mov                 cx, ax  ; save original FLAGS
    and                 ax, 0fffh ; clear bits 12-15 in FLAGS
    push                 ax      ; save new FLAGS value on stack
    popf                ; replace current FLAGS value
    pushf                ; get new FLAGS
    pop                 ax      ; store new FLAGS in AX
    and                 ax, 0f000h ; if bits 12-15 are set, then CPU
    mov                 ax, 0f000h ; is an 8086/8088
    cmp                 cpu_type, 0 ; turn on 8086/8088 flag
    je                 end_get_cpuid ; jump if CPU is 8086/8088

;
; Intel 286 CPU check
; Bits 12-15 of the FLAGS register are always clear on the
; Intel 286 processor in real-address mode.
;
check_80286:
    or                 cx, 0f000h ; try to set bits 12-15
    push                 cx      ; save new FLAGS value on stack
    popf                ; replace current FLAGS value
    pushf                ; get new FLAGS
    pop                 ax      ; store new FLAGS in AX
    and                 ax, 0f000h ; if bits 12-15 clear, CPU=80286
    mov                 cpu_type, 2 ; turn on 80286 flag
    jz                 end_get_cpuid ; if no bits set, CPU is 80286

;
; Intel386 CPU check
; The AC bit, bit #18, is a new bit introduced in the EFLAGS
; register on the Intel486 DX CPU to generate alignment faults.
; This bit cannot be set on the Intel386 CPU.
;
check_80386:
; It is now safe to use 32-bit opcode/operands
    mov                 bx, sp      ; save current stack pointer to align
    and                 sp, not 3   ; align stack to avoid AC fault
    OPND32
    pushf                ; push original EFLAGS
    OPND32
    pop                 ax      ; get original EFLAGS
    OPND32
    mov                 cx, ax      ; save original EFLAGS
    OPND32 35h, 40000h           ; flip (XOR) AC bit in EFLAGS
    OPND32
    push                 ax      ; save new EFLAGS value on stack
    OPND32
    popf                ; replace current EFLAGS value
    OPND32
    pushf                ; get new EFLAGS
    OPND32
    pop                 ax      ; store new EFLAGS in EAX
    OPND32
    xor                 ax, cx      ; can't toggle AC bit, CPU=80386
    mov                 cpu_type, 3 ; turn on 80386 CPU flag
    mov                 sp, bx      ; restore original stack pointer
    jz                 end_get_cpuid ; jump if 80386 CPU
    and                 sp, not 3   ; align stack to avoid AC fault

```

Example 1. CPU Identification Procedure (Continued)

```

    OPND32
    push    cx
    OPND32
    popf    ; restore AC bit in EFLAGS first
    mov     sp, bx ; restore original stack pointer

; Intel486 DX CPU, Intel487 SX NDP, and Intel486 SX CPU check
; Checking for ability to set/clear ID flag (Bit 21) in EFLAGS
; which indicates the presence of a processor
; with the ability to use the CPUID instruction.
;
check_80486:
    mov     cpu_type, 4 ; turn on 80486 CPU flag
    OPND32
    mov     ax, cx ; get original EFLAGS
    OPND32 35h, 200000h ; flip (XOR) ID bit in EFLAGS
    OPND32
    push    ax ; save new EFLAGS value on stack
    OPND32
    popf    ; replace current EFLAGS value
    OPND32
    pushf   ; get new EFLAGS
    OPND32
    pop     ax ; store new EFLAGS in EAX
    OPND32
    xor     ax, cx ; can't toggle ID bit,
    je      end_get_cpuid ; CPU=80486

; Execute CPUID instruction to determine vendor, family,
; model and stepping.
;
check_vendor:
    mov     id_flag, 1 ; set flag indicating use of CPUID inst.
    OPND32
    xor     ax, ax ; set up input for CPUID instruction
    CPUID ; macro for CPUID instruction
    OPND32
    mov     word ptr vendor_id, bx ; setup to test for vendor id
    OPND32
    mov     word ptr vendor_id[+4], dx
    OPND32
    mov     word ptr vendor_id[+8], cx
    mov     si, offset vendor_id
    mov     di, offset intel_id
    mov     cx, length intel_id

compare:
    repe    cmpsb ; compare vendor id to "GenuineIntel"
    or      cx, cx
    jnz     end_get_cpuid ; if not zero, not an Intel CPU,

intel_processor:
    mov     intel_proc, 1

```

Example 1. CPU Identification Procedure (Continued)


```

cpuid_data:
    OPND32
    cmp             ax, 1      ; make sure 1 is a valid input
                                ; value for CPUID
    jl             end_get_cpuid ; if not, jump to end
    OPND32
    xor             ax, ax      ; otherwise, use as input to CPUID
    OPND32
    inc            ax          ; and get stepping, model and family
    CPUID
    mov            stepping, al
    and            stepping, STEPPING_MASK ; isolate stepping info
    and            al, MODEL_MASK ; isolate model info
    shr            al, MODEL_SHIFT
    mov            model, al

    and            ax, FAMILY_MASK ; mask everything but family
    shr            ax, FAMILY_SHIFT
    mov            cpu_type, al ; set cpu_type with family

    OPND32
    mov            feature_flags, dx save feature flag data

end_get_cpuid:
    ret
get_cpuid endp

;*****

get_fpuuid proc
;
;   This procedure determines the type of FPU in a system
;   and sets the fpu_type variable with the appropriate value.
;   All registers are used by this procedure, none are preserved.
;
;   Coprocessor check
;   The algorithm is to determine whether the floating-point
;   status and control words can be written to. If not, no
;   coprocessor exists. If the status and control words can be
;   written to, the correct coprocessor is then determined
;   depending on the processor id. The Intel386 CPU can
;   work with either an Intel287 NDP or an Intel387 NDP.
;   The infinity of the coprocessor must be
;   checked to determine the correct coprocessor id.

    fninit          ; reset FP status word
    mov            fp_status, 5a5ah ; initialize temp word to
                                ; non-zero value
    fnstsw          fp_status      ; save FP status word
    mov            ax, fp_status    ; check FP status word
    cmp            al, 0           ; see if correct status with
                                ; written
    mov            fpu_type, 0      ; no fpu present
    jne            end_get_fpuuid

```

Example 1. CPU Identification Procedure (Continued)

```

check_control_word:
    fnstcw    fp_status    ; save FP control word
    mov      ax, fp_status  ; check FP control word
    and      ax, 103fh      ; see if selected parts
                                ; looks OK
    cmp      ax, 3fh        ; check that 1's & 0's
                                ; correctly read

    mov      fpu_type, 0
    jne      end_get_fpuuid
    mov      fpu_type, 1

;
; 80287/80387 check for the Intel386 CPU
;
check_infinity:
    cmp      cpu_type, 3
    jne      end_get_fpuuid
    fldl                     ; must use default control from FNINIT
    fldz                     ; form infinity
    fdiv                     ; 8087 and Intel287 NDP say +inf = -inf
    fld      st              ; form negative infinity
    fchs                     ; Intel387 NDP says +inf <> -inf
    fcompp                     ; see if they are the same and remove them
    fstsw    fp_status      ; look at status from FCOMPP
    mov      ax, fp_status
    mov      fpu_type, 2    ; store Intel287 NDP for fpu type
    sahf                     ; see if infinities matched
    jz      end_get_fpuuid  ; jump if 8087 or Intel287 is present
    mov      fpu_type, 3    ; store Intel387 NDP for fpu type
end_get_fpuuid:
    ret
get_fpuuid endp

;*****

print proc
;
; This procedure prints the appropriate cpuid string and
; numeric processor presence status. If the CPUID instruction
; was supported, this procedure prints out cpuid info.
; All registers are used by this procedure, none are preserved.

    cmp      id_flag, 1      ; if set to 1, cpu supports
                                ; CPUID instruction
                                ; print detailed CPUID information
    je      print_cpuid_data
    mov     dx, offset id_msg print initial message
    mov     ah, 9h
    int     21h

print_86:
    cmp      cpu_type, 0

```

Example 1. CPU Identification Procedure (Continued)

```

jne      print_286
mov      dx, offset c8086
mov      ah, 9h
int      21h
cmp      fpu_type, 0
je       end_print
mov      dx, offset fp_8087
mov      ah, 9h
int      21h
jmp      end_print

print_286:
cmp      cpu_type, 2
jne      print_386
mov      dx, offset c286
mov      ah, 9h
int      21h
cmp      fpu_type, 0
je       end_print
mov      dx, offset fp_80287
mov      ah, 9h
int      21h
jmp      end_print

print_386:
cmp      cpu_type, 3
jne      print_486
mov      dx, offset c386
mov      ah, 9h
int      21h
cmp      fpu_type, 0
je       end_print
cmp      fpu_type, 2
jne      print_387
mov      dx, offset fp_80287
mov      ah, 9h
int      21h
jmp      end_print

print_387:
mov      dx, offset fp_80387
mov      ah, 9h
int      21h
jmp      end_print

print_486:
cmp      fpu_type, 0
je       print_Intel486sx
mov      dx, offset c486
mov      ah, 9h
int      21h
jmp      end_print

print_Intel486sx:
mov      dx, offset c486nfp
mov      ah, 9h
int      21h
jmp      end_print

```

Example 1. CPU Identification Procedure (Continued)

```

print_cpuid_data:

cmp_vendor:
    cmp     intel_proc, 1
    jne     not_GenuineIntel

    cmp     cpu_type, 4           ; if cpu_type=4, print
                                ; Intel486 CPU message
    jne     check_Pentium
    mov     dx, offset Intel486_msg
    mov     ah, 9h
    int     21h
    jmp     print_family

check_Pentium:
    cmp     cpu_type, 5           ; if cpu_type=5, print
                                ; Pentium processor message
    jne     print_features
    mov     dx, offset Pentium_msg
    mov     ah, 9h
    int     21h

print_family:
    mov     dx, offset familymsg   ; print family msg
    mov     ah, 9h
    int     21h
    mov     al, cpu_type
    mov     byte ptr dataCR, al
    add     byte ptr dataCR, 30h    ; convert to ASCII
    mov     dx, offset dataCR      ; print family info
    mov     ah, 9h
    int     21h

print_model:
    mov     dx, offset modelmsg    ; print model msg
    mov     ah, 9h
    int     21h
    mov     al, model
    mov     byte ptr dataCR, al
    add     byte ptr dataCR, 30h    ; convert to ASCII
    mov     dx, offset dataCR      ; print model info
    mov     ah, 9h
    int     21h

print_stepping:
    mov     dx, offset steppingmsg ; print stepping msg
    mov     ah, 9h
    int     21h
    mov     al, stepping
    mov     byte ptr dataCR, al
    add     byte ptr dataCR, 30h    ; convert to ASCII
    mov     dx, offset dataCR      ; print stepping info
    mov     ah, 9h
    int     21h

print_features:
    mov     ax, feature_flags
    and     ax, FPU_FLAG           ; check for FPU

```

Example 1. CPU Identification Procedure (Continued)


```

        jz      check_MCE
        mov     dx, offset fpu_msg
        mov     ah, 9h
        int     21h

check_MCE:
        mov     ax, feature_flags
        and     ax, MCE_FLAG           ; check for MCE
        jz      check_CMPXCHG8B

        mov     dx, offset mce_msg
        mov     ah, 9h
        int     21h

check_CMPXCHG8B:
        mov     ax, feature_flags
        and     ax, CMPXCHG8B_FLAG     ; check for CMPXCHG8B
        jz      end_print
        mov     dx, offset cmp_msg
        mov     ah, 9h
        int     21h
        jmp     end_print

not_GenuineIntel:
        mov     dx, offset not_Intel
        mov     ah, 9h
        int     21h

end_print:
        ret

print endp

        end     start

```

Example 1. CPU Identification Procedure (Continued)



AB-53

APPLICATION BRIEF

The 85C224 Clock Driver Low Output Skew PLD

3

LISA J. LAU
COMPONENT RESEARCH AND DEVELOPMENT

JOHN VAN SACK
SALES AND MARKETING

PROGRAMMABLE LOGIC DEVICES

October 1993

The 85C224 Clock Driver Low Output Skew PLD

CONTENTS

PAGE

1.0 INTRODUCTION	3-361
2.0 CLOCK DISTRIBUTOR	3-361
3.0 FREQUENCY DIVIDER	3-362
4.0 INVERTED OUTPUTS	3-362
5.0 RISE AND FALL TIMES	3-362

CONTENTS

PAGE

6.0 TEMPERATURE DEPENDENCE ...	3-363
7.0 POWER SAVINGS	3-363
8.0 TOOLS AND SUPPORT	3-363
9.0 CONCLUSION	3-363

1.0 INTRODUCTION

With the continuing demand for performance driving the need for high frequency microprocessors and information systems (i.e., the Pentium™ processor, i486 microprocessors, and PCI Bus), designers of these complex systems are faced with the task of completing numerous system instructions within a shrinking window of time; from a 30 ns clock period @33 MHz to 15 ns @66 MHz. Many recent logic devices feature faster propagation delays, but this is not always enough. Designers are now seeking relief from these timing constraints by using higher quality clock driver devices, as well. At the current stage of advanced processor system development, clock signal timing accuracy is more important than ever. Designers must be able to drive clocks to various regions of the board using special clock driver chips featuring extremely low output pin-to-output pin skew (T_{OS}).

Intel programmable logic devices are a highly effective, low cost solution to the output skew minimization problem associated with high frequency clock distribution. The 85C224 Programmable Logic Device is designed with extremely low output pin skew. In addition, it offers superior output signal quality including fast rise and fall times. Combined with the flexibility of programmable logic the 85C224 can be programmed in a variety of clock driver configurations with maximum output pin skew of less than 400 ps.

Unlike dedicated clock drivers that have limited flexibility, Intel's 85C224 can be configured into different types of clock drivers. This flexibility allows the designer to satisfy several clock driver needs with one device. In addition, while dedicated clock drivers can perform only one function (typically distribution or division), the flexible 85C224 satisfies programmable logic needs such as control signals and widespread glue logic needs. Only with a minimized output skew PLD like the 85C224 can a designer implement clock distribution, division, and programmable logic with a single device.

Not only is the 85C224 flexible, but it is also cost effective. In comparison, dedicated clock driver devices with equivalent output skew range in cost of up to 10 times more than the 85C224! If the extrinsic value of the flexible programmable logic features were included in this cost comparison, the cost savings multiple would be even greater!

For power intensive systems, like notebook PCs, low power consumption clearly becomes important. Fortunately, the 85C224 has always incorporated low power CMOS technology (typical $I_{CC} = 60$ mA). The 85C224 is an ideal clock driver for low power notebook PC's and portable systems.

This Application Brief describes how the flexibility and extremely low output skews of the 85C224 meet the

clock driver needs of high clock speed systems. Two types of clock drivers are discussed, the simple gate driver or distributor and the frequency divider.

2.0 CLOCK DISTRIBUTOR

The 85C224 PLD performs as a simple gate driver or clock signal distributor when its outputs are programmed to replicate an input clock. In this configuration, the input clock signal is replicated and driven from between 1 to 8 output pins (see Figure 1).

From the outputs, the clock lines are then distributed to numerous regions of the system layout which require a synchronous version of the system clock as an input. Each of the output clocks has the same frequency and duty cycle as the original input clock. Output pin skew in this mode will typically be less than 250 ps for the low to high transition and 130 ps for the high to low transition (see Table 1). This configuration will drive/distribute clocks up to a maximum clock frequency of 133 MHz. A simple PLDasm program is shown below where the outputs (IO's) follow the clock input (inp1):

EQUATIONS

```
io1 = inp1
io1.TRST = VCC
io2 = inp1
io2.TRST = VCC
```

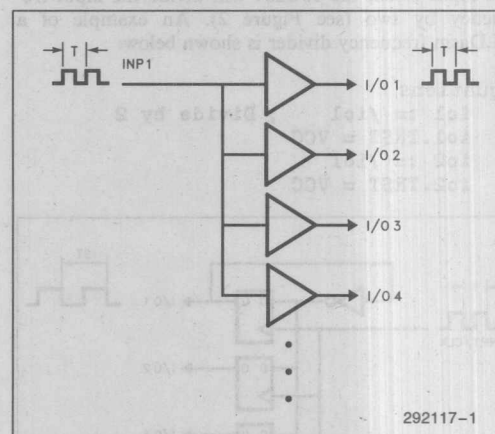


Figure 1. Gate Driver

Design Recommendations: The lowest output skew can be obtained by defining Input 6 (pin 6) as the clock input of choice. This path has been determined by engineering to provide the lowest output pin skew.

The best way to ensure a clean output clock with limited ground bounce is to connect all unused input pins and optional ground pins to a clean ground plane. In addition, connect all optional power pins to a clean power supply.

limited with a configuration driving all 8 outputs and measuring worst case skew between all combinations of outputs. If less than eight output clocks are needed then the designer can attain even better output skew by selecting the optimal output pins which maintain progressively better output skew characterizations. For the low to high transition, the output pin skew decreases as used clock outputs are closer to V_{CC} , pin 28. In addition, unused output pins should be tied to ground. For example, a 1 to 4 clock distribution using the first four I/Os (and grounding I/Os 5-8) provides output pin skew as low as 200 ps across all four outputs.

Conversely, for the high to low transition the skew increases as you move away from the ground pin. By grounding I/O1 and I/O8 the skew for the remaining pins are reduced to 100 ps. For systems requiring the least skew possible (≤ 100 ps), triggering from the falling clock edge will provide the smallest clock skew achievable by the 85C224 (or any device on the market).

3.0 FREQUENCY DIVIDER

For systems that require multiple clock frequencies (i.e., 66 MHz and 33 MHz) or a clock signal with 50% duty cycle, a clock frequency divider can be used. By programming the outputs to toggle on the rising edge of the input clock, the 85C224 will divide the input frequency by two (see Figure 2). An example of a PLDasm frequency divider is shown below:

Equations

```
io1 := /io1 ; Divide by 2
io1.TRST = VCC
io2 := /io1
io2.TRST = VCC
```

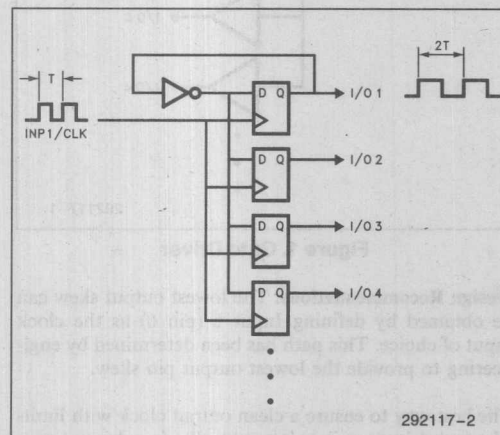


Figure 2. Frequency Divider

skews of 130 ps for the high to low transition and 200 ps for the low to high transition (see Table 1). These superior output skews are achieved as a result of the close proximity of the toggling registers to the output pin. The reduced signal path in the divider configuration translates into a narrowing of the potential variation in the output pin skew; hence, even less output pin skew.

The maximum frequency of the 85C224-100 clock divider is 100 MHz input with a 50 MHz output. The 85C224-10 will divide clock inputs up to 58 MHz and is offered at an even lower price.

As with the clock distributor, the best way to insure a clean output clock signal and limited ground bounce is to connect all unused input pins, other than the clock input pin, (INPUT 1) to a clean ground plane. Also, connect the pins labeled "no connection" to ground and Pin 1 to V_{CC} . The skew increases for I/Os as they move farther from the V_{CC} pin. Conversely, for the high to low transition the skew increases for outputs farthest from the ground pin. This information is often useful when routing low skew priority signals.

4.0 INVERTED OUTPUTS

In addition to low output pin skews and a variety of clock driver configurations the 85C224 can be programmed to invert any of the eight outputs. This feature reduces skew and saves parts as it eliminates the need to route a clock through an inverter chip to provide the capability of triggering on the falling edge of the clock. The inversion is done with a negligible increase in output skew. From a software standpoint, the inversion can be accomplished by programming the "invert" bit in the PLD source code. In PLDasm, inverting the signal is accomplished by using the active-low declaration "/" in front of the output signal that is to be inverted.

5.0 RISE AND FALL TIMES

The rise and fall times of the 85C224 meet the needs of high-speed system applications such as the Pentium™ microprocessor, while controlling ground bounce. Typical rise times are under 1.4 ns and typical fall times are under 1000 ps. Rise and fall times were measured from 0.8V to 2.0V with a 50 pF load.

6.0 TEMPERATURE DEPENDENCE

As the temperature approaches 85°C, the output skew of the 85C224 remains under 500 ps. In combinatorial logic, the output skew will increase to typical values of 350 ps for the low to high transition and 230 ps for the high to low transition. In registered mode the skew at 85°C is typically 200 ps for the low to high transition and 250 ps for the high to low transition.

7.0 POWER SAVINGS

The 85C224 is perfect for laptop system as the 1-micron CHMOS IIIE EPROM process allows it to consume less power than bipolar PALs and even CMOS GALS. Since there is an output enable P-term for every macro cell, a power management scheme can be programmed into the array that will disable macrocells according to the power requirements of the system. In this case, the 85C224 can perform three functions: power management, clock distribution and division, further demonstrating the high utility and flexibility of this device.

8.0 TOOLS AND SUPPORT

Complete design files and JEDEC files associated with this brief can be downloaded via the PLDshell Plus

BBS. The number is (916) 985-2308. You must be a registered PLDshell Plus user to log in.

PLDshell Plus software is available from Intel Literature 1-800-548-4725 or your local Intel sales office.

9.0 CONCLUSION

The 85C224 is an excellent choice for low skew clock driver applications. The PLD offers flexibility which allows the designer to configure the device into an assortment of clock driving implementations: distribution, division, and inversion. The device fundamentally provides low power and programmable logic functionality to meet the needs of control logic, state machines, and power management—all with a single device, the 85C224. The extremely low skews offered by the 85C224 are as good and better than comparable single function clock drivers; for a fraction of the cost!

High performance, low power, minimal output skew, programmable capabilities, low cost and overall flexibility make the 85C224 an intelligent solution for your design needs.

3

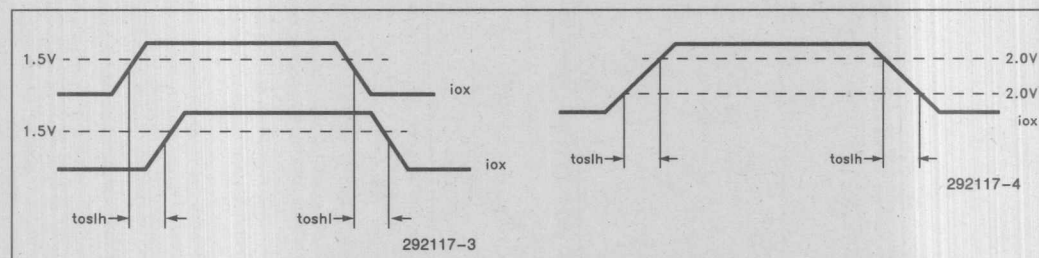


Figure 3. Measurements

Table 1. 85C224-10 Characteristic Data

Symbol	Parameter	V _{CC}	T _A = 25°C C _L = 50 pF		T _A = 75°C C _L = 50 pF		Units
			Typ	Max	Typ	Max	
Comb t _{OSHL}	Combinatorial I/O to I/O Skew High to Low Transition	5.0V	130	250	175	250	ps
Comb t _{OSLH}	Combinatorial I/O to I/O Skew Low to High Transition	5.0V	250	400	200	300	ps
Reg t _{OSHL}	Registered I/O to I/O Skew High to Low Transition	5.0V	150	225	150	225	ps
Reg t _{OSLH}	Registered I/O to I/O Skew Low to High Transition	5.0V	200	300	200	250	ps
t _r	Rise Time (0.8V–2.0V)	5.0V	1.3	1.5	1.2	1.4	ns
t _f	Fall Time (0.8V–2.0V)	5.0V	1.0	1.2	1.1	1.1	ns
f _{MAX}	Max. Counter Freq. 1/t _{CNT} Internal Feedback	5.0V	80	62.5	80	62.5	MHz
f _{MAX}	85C224-100 Internal Feedback 1/t _{CNT}	5.0V		115		115	MHz

